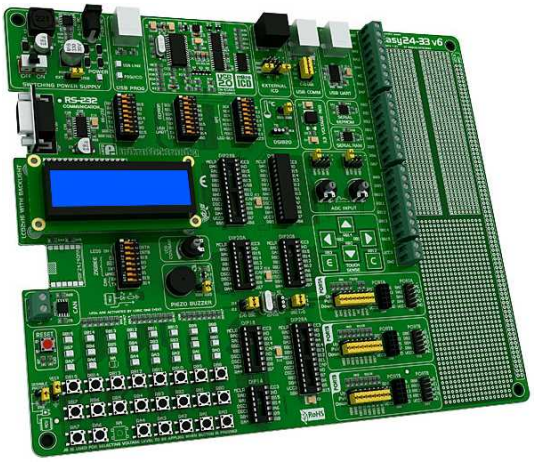
型番:dsPICF-300　　REV.1.00

USB2.0対応 高速プログラマ搭載
DIPタイプPIC24/dsPIC33シリーズ対応
開発用統合評価ボード

取扱説明書
お使いになる前にこの説明書をよくお読みの上正しくお使いください。

(C)2013 マイクロテクニカ

ボード本体



製品の概要

DIPタイプPIC24/dsPIC33シリーズ対応開発用統合評価ボード(型番:dsPICF-300、以下dsPICF-300と記載)は、ボード上にUSB2.0対応のPICプログラマーを搭載し、最新の16ビットマイコン、PIC24シリーズ及びDSP機能を搭載したdsPIC33シリーズに対応した統合開発ボードです。16ビットタイプのPIC24シリーズやdsPIC33シリーズはピン数の多い面実装タイプのものが多いですが本ボードでは、入手しやすく、かつ扱いやすいDIPタイプのデバイスに対応したボードです。

ボード上には、14ピン、18ピン、20ピン、28ピンのソケットがあり、ピン数の異なるPIC24/dsPIC33デバイスを搭載して開発することができます。また、ピンの配置が異なる28ピン及び20ピンのデバイスも装着できるよう、ソケットを用意しました。

ボード上に搭載のUSB接続マイコンライターは、現在流通しているほぼすべてのPIC24/dsPIC33シリーズに対応しており、基板上に装着したマイコンにプログラムを書き込みできます。USBバスから電源を取りますので、別途電源を準備する必要がありません。

ボードに搭載されている各周辺回路に接続されているピンは周辺回路に接続して利用するか又は周辺回路には接続しないで使用するかをディップスイッチやジャンパーソケットで選択できます。また全I/Oピンはボード右側に実装されているヘッダピンから取り出すことができます。また、本セットには6KワードまでのHEXファイルが生成できるICD機能付きの体験版Cコンパイラーを付属。本書にはチュートリアルも収録されているので、すぐに16ビットマイコンの世界を体験できます。

パッケージの内容

■同梱物
・dsPICF-300ボード本体
・PIC24F16KA102-I/SP （ボードに装着済み）
・16文字×2行液晶ディスプレイモジュール （ボードに装着済み）
・USBケーブル
・CD-ROM
・マニュアル(本書)

ドライバのインストール及びパソコンとの接続

■デバイスドライバーのインストール
dsPICF-300をパソコンと接続する前に、デバイスドライバーをパソコンにインストールします。
(Windows98及びMEをご使用のお客様は本項は読み飛ばして、次の「パソコンと接続する」の項よりお読み下さい。)
WindowsXP.Vista,7をご利用のお客様は下記の手順で、ドライバーをインストールします。

1 付属のCD-ROMをパソコンのCD-ROMドライブに挿入してエクスプローラー等で内容を表示します。

2 "USB Programmer Software"のフォルダを開きます。

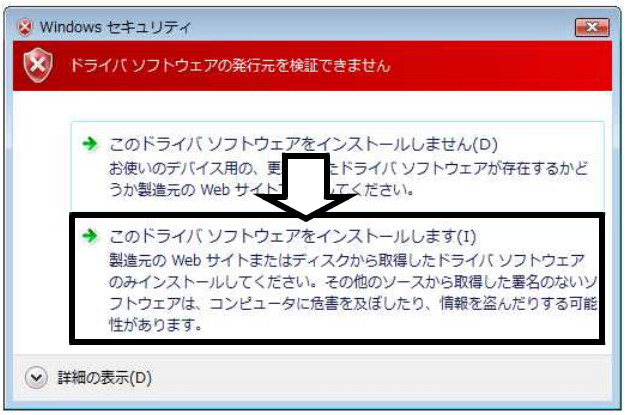
3 "Driver"フォルダを開きます。

4 OSごとにフォルダがあります。ご使用のパソコンのOSにあったフォルダを開きます。
※WindowsVista,7に関しては32ビット版用と64ビット版用がありますので間違えないようご注意ください。

5 開くと実行ファイル、"USB18PRG～.exe"がありますのでダブルクリックして実行します。

6 実行するとダイアログが表示されますので、そのまま"次へ"をクリックして続行します。

7 ドライバーのインストールが開始されます。
WindowsVista環境で下記のような警告ダイアログが表示された場合には、"このドライバソフトウエアをインストールします"をクリックして続行してください。



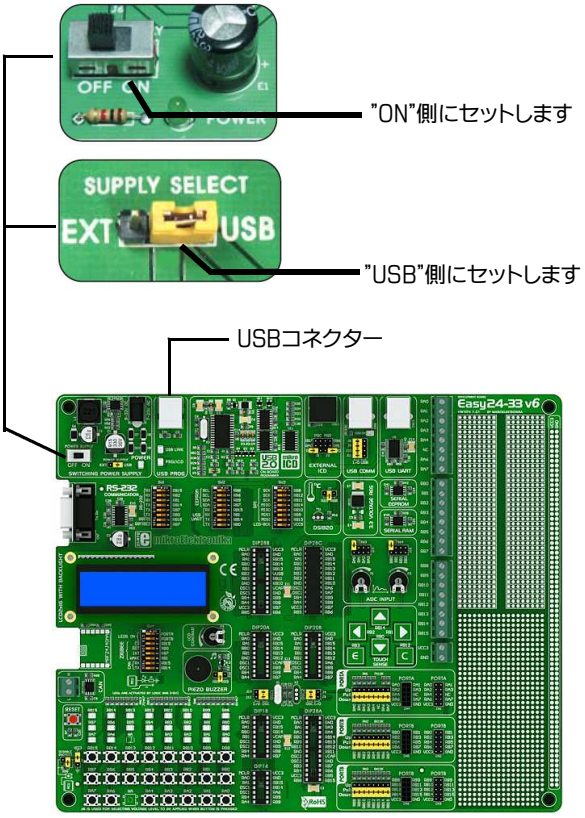
下図のようなメッセージが表示された場合には、"インストール"ボタンを押してください。

8 インストール作業が完了すると、下図のような画面になりますので、"完了"ボタンを押してインストールを完了します。
この時、"状態"のボックスに「使用できます」と表示されていることをご確認下さい。エラーが表示された場合には、ご使用OSの種類と、実行したデバイスドライバーの種類が一致しているかご確認下さい。

■パソコンと接続する

パソコンのUSBポートにdsPICF-300を接続します。

1 dsPICF-300の電源スイッチをONにします。
また、電源をUSBのバスパワーより給電しますので、"POWER SUPPLY"のジャンパー"USB"側にソケットを装着してください。



2 USBケーブルで、パソコンのUSBポートと、dsPICF-300を接続します。

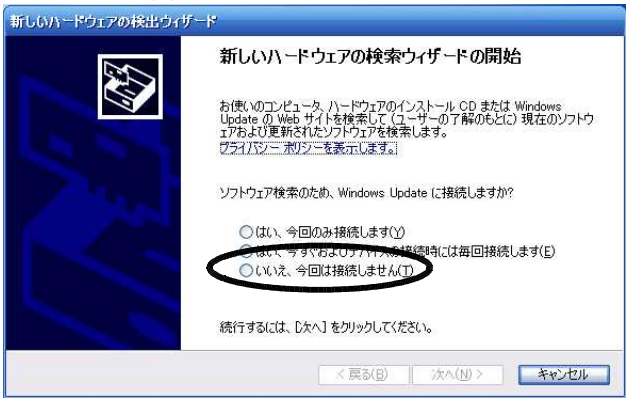
※パソコンのUSBポートがUSB2.0対応の場合には高速書き込みが行えます。USB1.1ポートの場合には通常速度での書き込みとなります。

→接続するとPOWERの緑LEDが点灯します。
→新しいハードウエアの検出ウィザードが自動的に起動します。

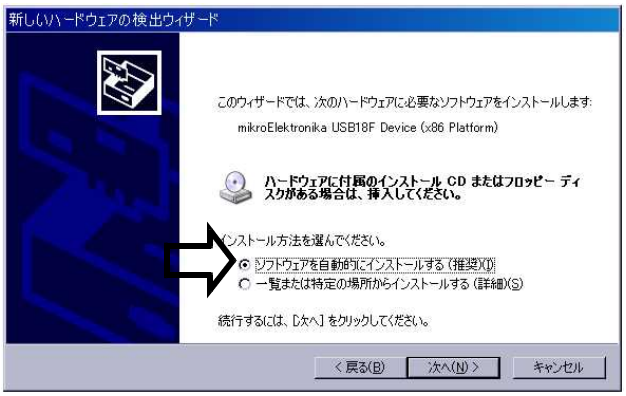
以下OSによって操作方法が異なります。

■Windows2000, XPをご利用の場合

3 下図のようなダイアログが表示されたら"いいえ、今回は接続しません"にチェックを入れて"次へ"ボタンをクリックします。



4 下記のようなダイアログが表示されますので、"ソフトウエアを自動的にインストールする(推奨)"にチェックをいれて、"次へ"ボタンをクリックします。



5 「ソフトウエアをインストールしています。お待ち下さい」というメッセージが表示されます。しばらくするとインストールが完了して下図のような表示となりますので"OK"ボタンをクリックして終了します。

■WindowsVista,7をご利用の場合----------

3 WindowsVista,7の場合には、dsPICF-300を接続すると自動的にデバイスドライバーのセットアップが開始されます。
あらかじめデバイスドライバーは先の手順にてインストールされていますので、WindowsVistaは自動的にデバイスドライバーを検出してインストールを完了します。

接続後しばらく待ちますと、下図のようなポップアップがタスクバーに表示されます。

上図のホップアップが表示されればインストールは完了です。

■Windows98(SE), MEをご利用の場合----------

3 自動的にデバイスが検出されます。
"新しいハードウエアの追加ウィザード"が表示されたら「次へ」を押して続行します。

4 次のダイアログでは,"使用中のデバイスに最適なドライバを検索する"にチェックを入れて,「次へ」をクリックします。

5 "検索場所の指定"にチェックを入れて,「参照」ボタンを押します。
ドライバの場所を指定するダイアログが表示されますのでCD-ROM内の"USB Programmer Software"フォルダ内の"Driver"フォルダにある"WIN98_ME"フォルダを指定してください。CD-ROMドライブがQドライブの場合にはディレクトリは下記の通りとなります。
Q:¥USB Programmer Softeare¥Driver¥WIN98_ME

指定したら「次へ」を押して続行します。
インストールが完了したら,「完了」ボタンを押して終了します。

～ここからは各OS共通の項目です～

dsPICF-300ボード上の"USB Link"の黄LEDが点灯していることを確認してください。
黄LEDの点灯はPC側にdsPICF-300が正しく認識されていることを示します。

※"USB Link"のLEDが点灯していない場合には、正しくドライバーがインストールされていません。再度手順を確認の上、ドライバーをインストールし直しておためしください。

※デバイスマネージャーに"USB Devices by mikroElektronika"が追加されます。

WIndowsVista及び7環境でご使用のお客様へ ユーザーアカウント制御無効設定のお願い

WindowsVista及び7環境で、dsPICF-300をご使用の場合、Windowsに搭載されたユーザーアカウント制御(UAC)機能がUSBポートへのアクセスを拒否することにより、下図のようなエラーメッセージが表示されて、書き込みができないという現象を確認しています。

この現象は、WindowsVista及び7において標準で有効になっているユーザーアカウント制御機能を無効に設定することで回避できます。下記の方法で設定を変更して頂けますようお願い致します。

※なおこのエラーは、WindowsVista及び7以外の環境でお使いのお客様には関係ありません。

■Windows Vistaのユーザーアカウント制御を無効に設定する

1 スタートボタンをクリックして、コントロールパネルを表示します。

2 コントロールパネルから"ユーザーアカウント"をダブルクリックします。"ユーザーアカウント"アイコンがない場合には、ウインドウ左上の"クラシック表示"をクリックして表示を変更します。

3 "ユーザーアカウント制御の有効化または無効化"のリンクをクリックします。

4 "ユーザーアカウント制御(UAC)を使ってコンピューターの保護に役立たせる"のチェックを外します。

5 "OK"ボタンを押して完了します。パソコンを再起動します。

■Windows7のユーザーアカウント制御を無効に設定する

1 スタートボタンをクリックして、コントロールパネルを表示します。

2 コントロールパネルから""ユーザーアカウントと家族のための安全設定"をクリックします。

3 "ユーザーアカウント"をクリックします。

4 "ユーザーアカウント制御設定の変更"をクリックします。

5 画面に表示されるスライドバーを一番下の"通知しない"に設定します。

6 "OK"ボタンを押して完了します。パソコンを再起動します。

ソフトウエアのインストール

開発に必要なソフトウエアをインストールします。なお、ソフトウエアをインストールする際には、必ずUSBポートからdsPICF-300を外した状態で行ってください。

■MPLABのインストール

MPLABはPICマイコンのプログラム開発用の統合開発環境です。PICマイコンの開発元マイクロチップ社から提供されています。MPLABがお手持ちのパソコンにインストールされていない場合には、インストールしてください。

1 CD-ROM内の"MPLAB"フォルダを開きます。
"SETUP.EXE"をダブルクリックして実行します。
"Next>"をクリックして続行します。

2 "License Agreement"のダイアログが表示されたら内容をよくご確認の上、同意する場合には"I accept the terms of the licence agreement"にチェックを入れて、"Next>"クリックして続行します。
なお同意しないとインストールは続行できません。

3 "setup type"の選択ダイアログではすべてのプログラムをインストールしますので"Complete"にチェックを入れて"Next>"をクリックします。

4 MPLABをインストールするディレクトリを指定します。
特に設定する必要がなければデフォルトのままにしておきます。
インストールするディレクトリを変更したい場合には,"Browse"ボタンを押してインストールディレクトリを設定します。
なお変更した場合には,インストールディレクトリをメモするなど忘れないようにしてください。
"Next>"をクリックして続行します。

5 以降のダイアログでは次のように進めます。

■Application Maestro License
→"I accept the terms of the license agreement"にチェックをいれて"Next>"をクリックします。

■MPLAB C32 License
→"I accept the terms of the license agreement"にチェックをいれて"Next>"をクリックします。

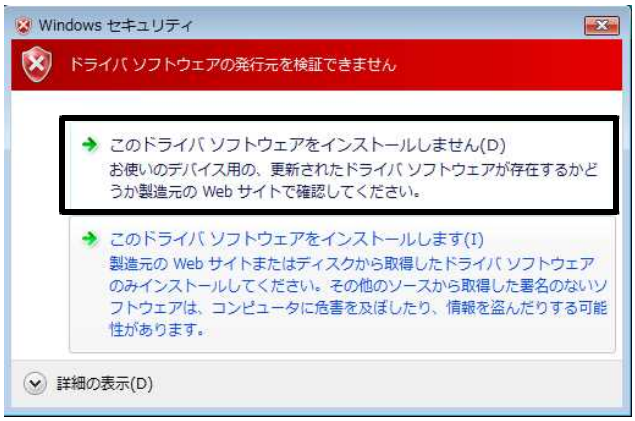
■Start copying files
→"Next>"を押します。
・・・インストールが開始されます。完了までしばらく待ちます。

6 インストールが完了すると、下図のようなダイアログが表示されます。MPLABバージョン8.00以降からはPICCコンパイラーのライトバージョンがMPLABに付属するようになったためです。
ここではOKボタンを押して続行します。

7 下図のようなPICC-Liteのインストール画面が表示されますので、ここでは"Cancel"ボタンを押してインストールをキャンセルします。

※PICC-Liteは、Hi-Tech社のPIC用Cコンパイラーです。MPLAB8.00以降ではライト版(機能限定版)が付属しました。インストールしていただいてもかまいませんが、本キットではこのソフトウエアは使用しません。

8 Windows Vista環境にインストールされている場合には、最後に下図のようなドライバーインストールに関するセキュリティ警告が表示されることがあります。
この場合には、"このドライバソフトウエアをインストールしません(D)"をクリックしてください。
ダイアログが消えたら"Finishボタンを押してインストールを完了します。



■mikroC PRO for dsPICのインストール

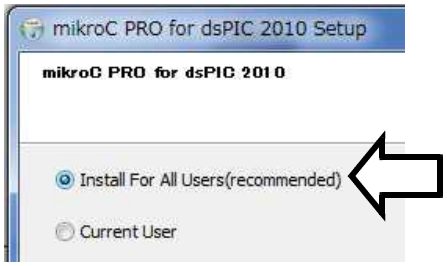
mikroElektronika社からリリースされているdsPICシリーズ用のANSI規格準拠のCコンパイラーです。

本dsPICF-300には、生成されるHEXファイルのコードが6KBまでに制限された体験版のCコンパイラーが付属しております。生成されるコードサイズに制限はありますが、使用できる機能及び期限等に制限はありません。この体験版Cコンパイラーをご使用になり、コードサイズの限定がない製品版をご希望の場合には、当方よりフル機能版を特別優待価格(10%引き価格)にて販売しております。

1 CD-ROM内の"Cコンパイラー"フォルダを開きます。
"SETUP.EXE"をダブルクリックして実行します。
"Next>"をクリックして続行します。

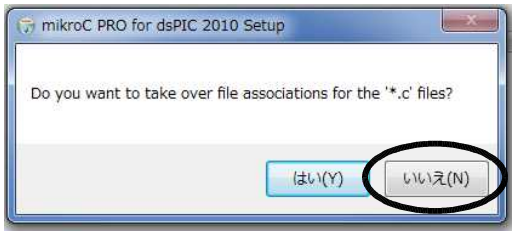
2 "License Agreement"のダイアログが表示されたら内容をよくご確認の上、同意する場合には"I accept the terms in the License Agreement"にチェックを入れて、"Next>"クリックして続行します。
なお同意しないとインストールは続行できません。

3 次の画面では、"Install For All Users(recommended")にチェックを入れて、"Next>"ボタンを押して続行します。



4 "Choose Components"ダイアログが開きますので、"□Examples"にチェックが入っていることを確認して、"Next"ボタンを押します。

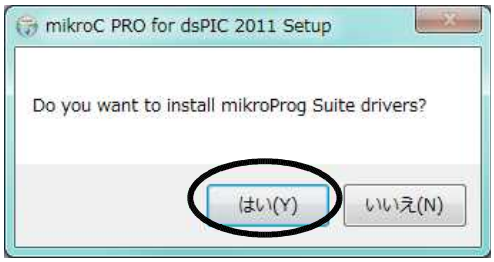
5 インストールするディレクトリを指定します。特に変更する理由がなければ、"Install"ボタンをクリックします。インストールが開始されます。インストール開始後下図のようなメッセージが表示されたら、"いいえ"をクリックして続行します。



6 "Completing the mikroC dsPIC Setup Wizard"というダイアログが表示されたら、"Finish"ボタンを押して終了します。
続いて、下記のようなメッセージが表示されますので、"はい"をクリックします。

7 再度、インストーラーが起動します。画面の指示に従って、インストールを進めてください。"License Agreement"のダイアログが表示されたら内容をよくご確認の上、同意する場合には"I accept the terms in the License Agreement"にチェックを入れて、"Next>"クリックして続行します。

8 インストールを完了すると再度、下図のようなメッセージが表示されます。"はい"を選択して、デバイスドライバーもハードディスクに配置しておきます。



■PICプログラマ用のソフトウエア

PICへプログラムを書き込む際に使用するプログラマソフトウエアです。mikroCをインストールした際にはmikroProg Suite For PICとして書き込みソフトウエアはインストールされていますが、mikroCをインストールしなかった場合には、書き込みソフトウエア単体でのインストールが必要です。
mikroCをインストールした場合には、この手順は飛ばしてください。

1 CD-ROM内の"USB Programmer Software"内にある"SETUP.EXE"をダブルクリックして実行します。インストーラーが起動しますので、"Next"をクリックして続行します。

2 ライセンス許諾画面が表示されますので、同意される場合には、"I accept the terms in the License Agreement"の方にチェックを入れて、"Next"ボタンをクリックします。

3 次の画面はそのまま"Next"をクリックします。

4 インストールするディレクトリを指定します。mikroCをご使用になる場合には、必ずmikroCをインストールしたドライブと同じドライブにインストールしてください。(例えばmikroCをCドライブにインストールした場合には、こちらのソフトウエアもCドライブにインストールしてください。)
"Install"ボタンを押すとインストールが開始されます。

5 インストールが完了したら"Finish"ボタンを押して完了します。

電源の投入方法

dsPICF-300は、USB接続にてUSBバスから電源の供給を受けることができます。この場合、別途電源を供給する必要はありません。

しかしdsPICF-300側でおよそ300mA以上を消費する予定がある場合にはUSBバスパワー給電では電力が不足する場合があります。この場合には別途外部からACアダプタにて電源を供給する必要があります。給電方法はジャンパーソケットによって設定を行います。
なお工場出荷時の設定はUSBバスパワー給電の設定になっています。

■USBバスパワーから給電する場合
dsPICF-300ボード上の"POWER SUPPLY"ジャンパーピンを「USB」側にショートします。

■ACアダプタから給電する場合
dsPICF-300ボード上の"POWER SUPPLY"ジャンパーピンを「EXT」側にショートします。

※ACアダプタは出力電圧がDC9Vで500mA以上のものをご用意ください。径は2.1φです。(当方にて専用ACアダプタを販売中です)

■電源スイッチ
dsPICF-300には、電源スイッチが装着されています。このスイッチは、給電方法に関係なく、ボードのスイッチのON/OFFができます。

対応デバイス

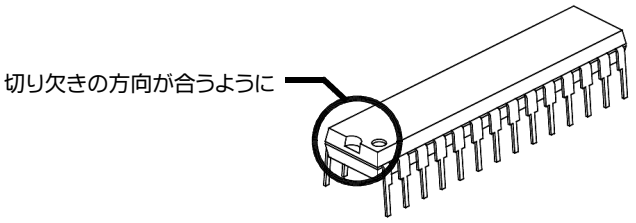
dsPIC33FJ12GP201-DIP18, dsPIC33FJ12MC201-DIP20
dsPIC33FJ06GS102-DIP28 , dsPIC33FJ06GS202-DIP28
dsPIC33FJ16GS402-DIP28 , dsPIC33FJ16GS502-DIP28
dsPIC33FJ12GP202-DIP28 , dsPIC33FJ12MC202-DIP28
dsPIC33FJ32GP202-DIP28 , dsPIC33FJ128GP802-DIP28
dsPIC33FJ128GP202-DIP28 , dSPIC33FJ64GP802-DIP28
dsPIC33FJ64GP202-DIP28 , dsPIC33FJ32GP302-DIP28
dsPIC33FJ32MC202-DIP28 , dsPIC33FJ128MC802-DIP28
dsPIC33FJ128MC202-DIP28 , dsPIC33FJ64MC802-DIP28
dsPIC33FJ64MC202-DIP28 , dsPIC33FJ32MC302-DIP28
PIC24F04KA200-DIP14 , PIC24F04KA201-DIP20
PIC24F08KA101-DIP20 , PIC24F16KA101-DIP20
PIC24F16KA102-DIP20 , PIC24F08KA102-DIP20
PIC24FJ16GA002-DIP28 , PIC24FJ32GA002-DIP28
PIC24FJ48GA002-DIP28 , PIC24FJ64GA002-DIP28
PIC24FJ32GA102-DIP28 , PIC24FJ64GA102-DIP28
PIC24FJ32GB002-DIP28 , PIC24FJ64GB002-DIP28
PIC24HJ12GP201-DIP18 , PIC24HJ12GP202-DIP28
PIC24HJ32GP202-DIP28 , PIC24HJ128GP802-DIP28
PIC24HJ128GP202-DIP28 , PIC24HJ64GP802-DIP28
PIC24HJ64GP202-DIP28 , PIC24HJ32GP302-DIP28

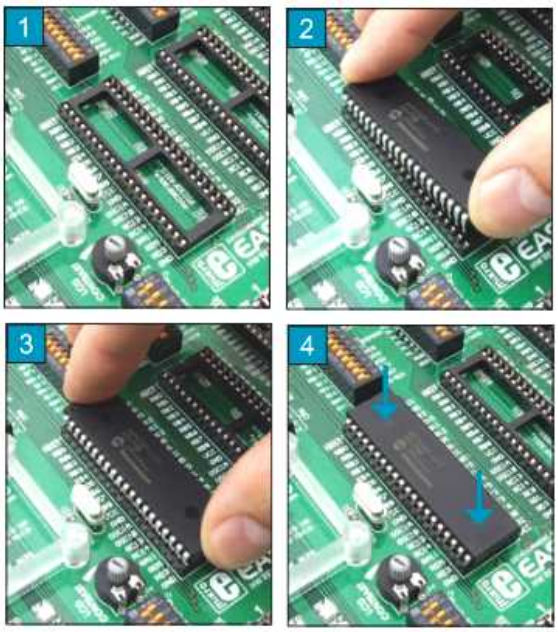
※上記のデバイスのDIPタイプのデバイスに対応しています。

PICマイコンの装着

使用するPICマイコンを、dsPICF-300ボード上のICソケットに装着します。工場出荷時は、PIC24F16KA102-I/SPがすでに装着されています。

装着できるPICマイコンは、必ず1種類だけです。複数のソケットにマイコンを装着すると故障の原因となります。装着時には、装着方向を間違わないようにICソケットの切り欠き部分とdsPICマイコンの切り欠き部分が合うように正しく装着してください。



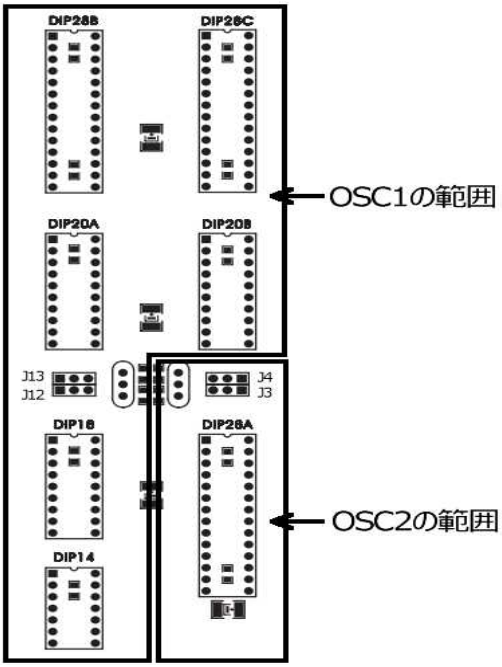


■20ピン及び28ピンデバイスを装着する場合
20ピン及び28ピンのマイコンでは、型式によってピンの配置が異なっています。dsPICF-300では、20ピンソケットを2つと、28ピンソケットを3つ用意しており、ピン配置の異なるデバイスでも装着できるように設計されています。

装着するデバイスのピン配置をデータシートにてご確認頂き、基板上のソケット位置に印刷されているシルク印刷と一致するソケットに装着してください。ソケットを間違えると正しくマイコンが動作しない他、場合によってはマイコンを破損してしまうことがありますので、装着前に必ずご確認ください。

マイコンの種類によって外部の水晶発振子又はセラミック発振子を取り付けて使用します。内蔵発振子のものは、外部発振子は取り外した状態でそのピンをI/Oピンとして使用できます。その場合には、J13・J12、J4・J3をOSC側からI/O側に設定してご使用ください。詳しくは次の通りとなります。

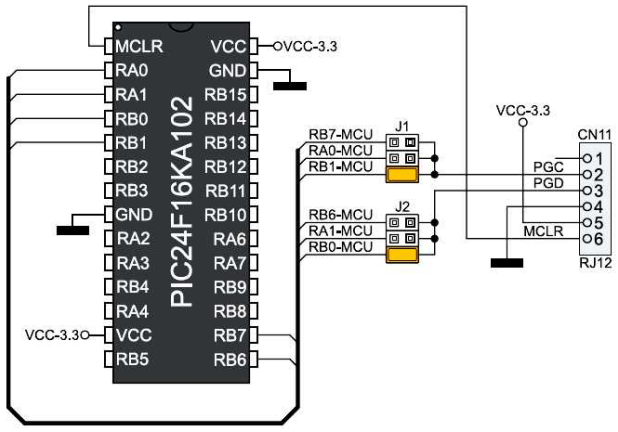
オシレータ端子のOSC1及びOSC2、それらの設定ジャンパーについては下図の通りになっています。



ジャンパー	ポジション	機能概要
J13	I/O	RA3がI/Oピンに設定されます
	OSC	OSC1のピンがRA3ピンに接続されます。外部発振子を使う時に設定します。
J12	I/O	RA2がI/Oピンに設定されます
	OSC	OSC1のピンがRA2ピンに接続されます。外部発振子を使う時に設定します。
J4	I/O	RB2がI/Oピンに設定されます
	OSC	OSC2のピンがRB2ピンに接続されます。外部発振子を使う時に設定します。
J3	I/O	RB1がI/Oピンに設定されます
	OSC	OSC2のピンがRB1ピンに接続されます。外部発振子を使う時に設定します。

MPLAB-ICD2及びICD3を使用する場合

dsPICF-300には、マイクロチップ社純正のMPLAB-ICD2又は、MPLAB-ICD3を接続するための6ピンモジュラーコネクターが付いています。ここにMPLAB-ICD2又はICD3を接続することでdsPICF-300をターゲットボードとして使用することができます。
内部の結線は下記のようになっています。



PGCピンと、PGDピンの結線は、使用するデバイスに応じてジャンパーソケットJ1及びJ2にて設定できます。デバイスのデータシートを参照し、ジャンパーソケットにて適切なピンに設定してください。

■MPLAB-ICD2及びICD3の設定

MPLAB-ICD2・ICD3のVddピンはdsPICF-300のVddと接続されています。MPLAB-ICD2・ICD3接続して、dsPICF-300をターゲットボードとして使用する場合には、電源の設定方法及びMPLAB-ICD2・ICD3の電源設定を下記のように設定してください。接続及び設定を誤ると、機器が故障する原因となります。

・dsPICF-300の電源は必ずACアダプタにて外部電源から給電する
・dsPICF-300のUSBはパソコンと接続しない
・dsPICF-300のUSBライター機能は使用しない
・MPLABで電源の設定を、"Target has own power supply"にする

ICD2・ICD3では、ターゲットボード側にICD2・ICD3側から電源を供給するモードと、ターゲットボードとICD2・ICD3の電源を別々にするモード、2つのモードが選択できます。ICD2・ICD3を利用する場合には、必ず後者のターゲットボードとICD2・ICD3の電源を別々のするようにします。(dsPICF-300の電源給電はdsPICF-300に接続したACアダプタで行い、ICD2・ICD3の電源給電は、USBバスパワーで行う)
下記の手順で電源の設定を行ってください。

1 dsPICF-300とICD2・ICD3は接続していない状態で、MPLABを起動し、メニューバーの"Debugger"又は"Programmer"をクリックして、使用するツールとして"MPLAB ICD2"又は"MPLAB ICD3"を選択し使用できるようにします。

2 再度メニューバーの"Debugger"又は"Programmer"をクリックして、"Settings"をクリックします。

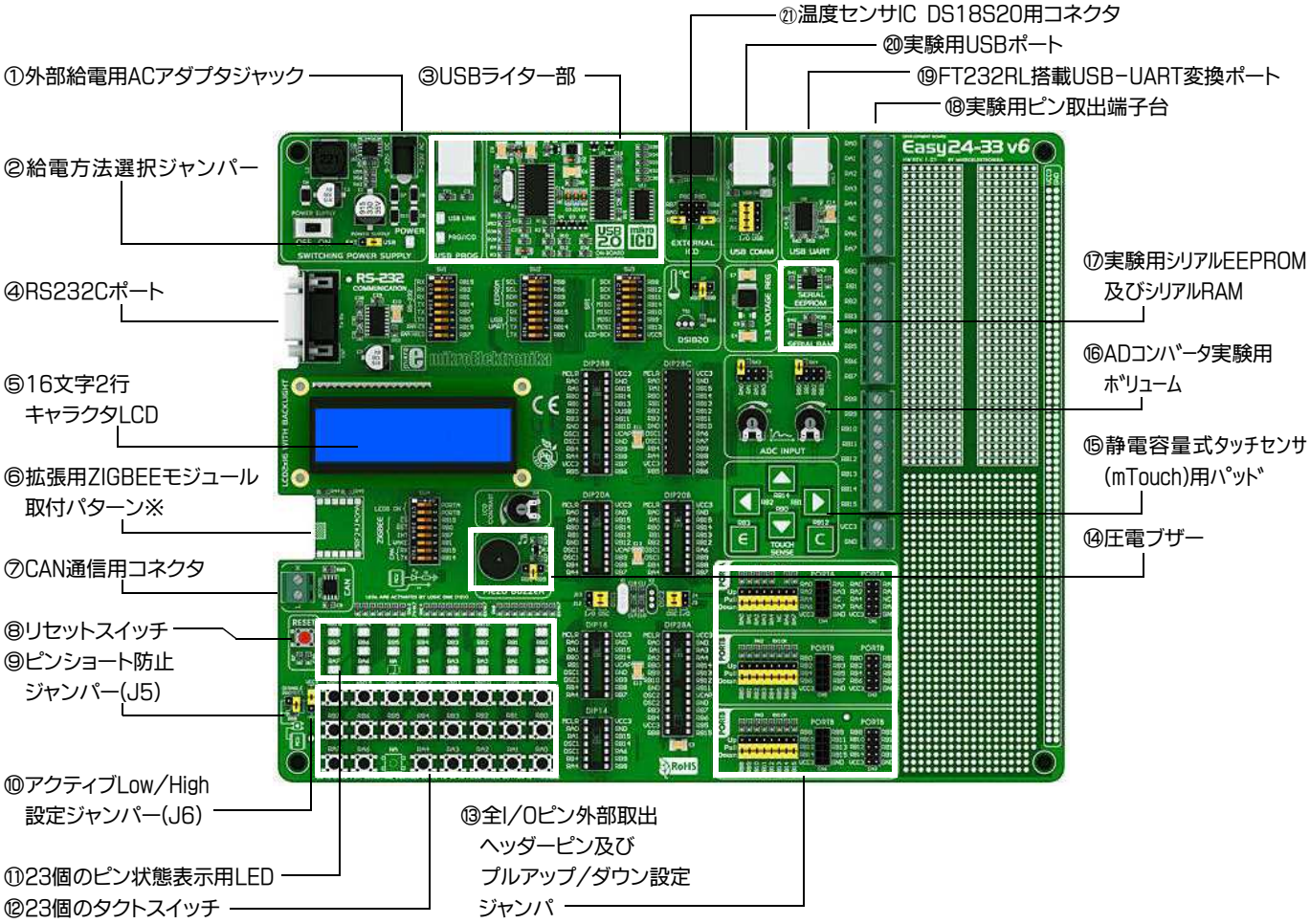
3 表示されたダイアログの中で"Power"タブをクリックします。

4 "Power target circuit from MPLAB ICD2(又はICD3)"のチェックを外します。無効になっている場合には、チェックが入っていないことを確認してOKボタンを押します。

5 dsPICF-300に電源を投入し、J1及びJ2のジャンパーでPGD及びPGCのピンを設定します。
ICD2・ICD3の電源が入っていることを確認して、6ピンモジュラーケーブルで、dsPICF-300のモジュラーソケットに接続します。

設定を正しく行ってご使用ください。MPLAB ICD2又はICD3を使用する場合には、絶対にdsPICF-300に搭載のUSBライター機能は使用しないでください。dsPICF-300に搭載のUSBライター機能を使用すると、両製品が破損します。この誤りを防ぐため、MPLAB ICD2又はICD3を使用する場合には、dsPICF-300のUSBポートには何も接続せず、給電は必ずACアダプタから行うようにしてください。

dsPICF-300の各部の説明



※ZIGBEEモジュールパターンは将来の拡張用です

---

①ACアダプタジャック及び電源セレクトジャンパ

dsPICF-300の電源をACアダプタから給電する場合には、このジャックにDC9V出力で2.1φのACアダプタを接続します。センター極性は問いません。電流は500mA以上を取り出せる安定化されたものを使用してください。(センター極性は問いません)

※給電方法については、②のジャンパー設定が必要です。

②給電方法選択ジャンパー

dsPICF-300への給電方法を選択します。USBパスパワーから給電するのか、ACアダプタより給電するのかをジャンパーピンで設定します。

③オンボードUSBライター、USBコネクタ

オンボードUSB2.0マイコンライター部です。

パソコン本体のUSBポートに接続します。付属の専用CコンパイラーmikroCと組み合わせることで、ICD機能を実現することができます。

USBデバイスとして正しくパソコンに認識されると、USBコネクタすぐ下の"USB LINK"のLEDが点灯します。プログラム書き込み中やICD実行中は"PRG/ICD"のLEDが点灯します。

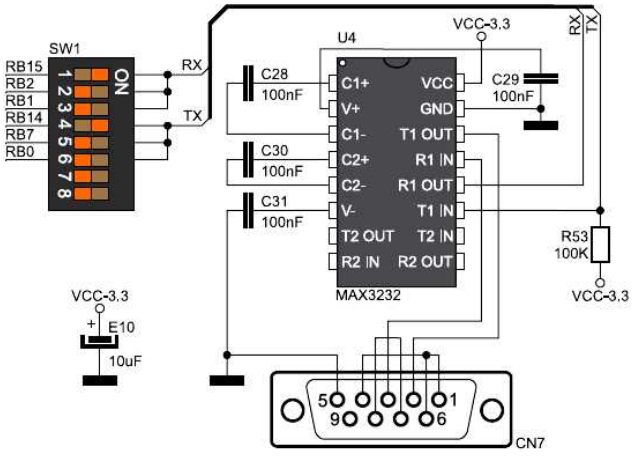
④RS232C通信ポート

レベル変換IC搭載のRS232Cポートです。

RX(受信データ)及びTX(送信データ)の信号線の接続先は、SW1のディップスイッチによって、選択することができます。

RS232CのTX線とRX線をそれぞれマイコンのどのピンと接続するかは、SW1部のシルク印刷を見ながら設定を行います。

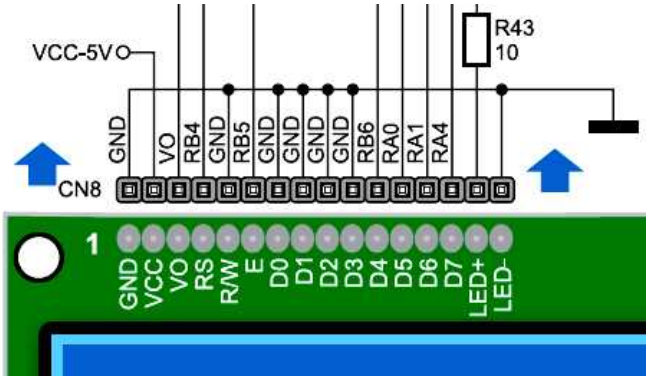
RS232C通信の実験をする場合には、パソコンとD-SUB9ピンの全結線ストレートケーブルをご使用下さい。



⑤16文字2行キャラクタ液晶ディスプレイ

2行16桁のLCDです。データ線にD7～D4の4ビットを使用する4ビットモードで使用できます。コントラスト調節は、P4ボリュームで行うことができます。

LCDのデータピンD4～D7はマイコンのRB6,RA0,RA1,RA4に、EnableビットはRB5に、RSピンはRB4に接続されています。



なお、LCDの各ピンと接続されているマイコンのピンをI/Oピンとして使用しようとすると、LCDを取り付けたままの場合、LCD側に電流が流れ、電圧レベルが正しくOV-Vccで出力されません。

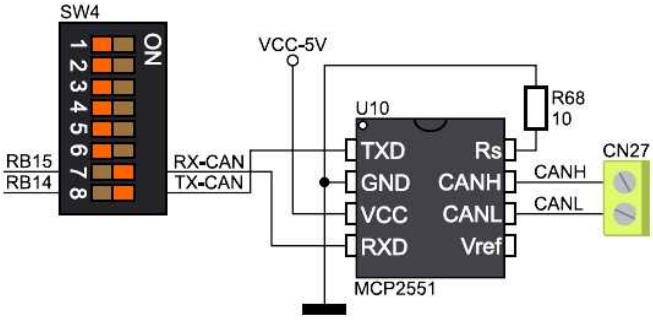
これらのピンをI/Oピンとして使用する場合には、LCDを取り外してご使用ください。

⑦CAN通信用コネクタ

CAN通信を行うための信号線、CANHとCANLがそれぞれ接続された端子台が設置されています。

CANコントローラには、MCP2551が接続されており、RXピン及びTXピンは、ディップスイッチSW4を介して、RB15とRB14にそれぞれ接続されるようになっています。

CAN通信機能を使用する場合には、SW4の7及び8スイッチをONにします。

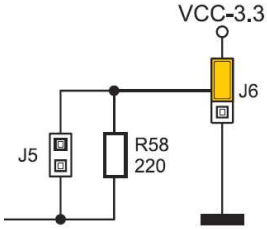


⑧リセットスイッチ

マイコンの$\overline{MCLR}$ピンと接続されており、スイッチを押すとマイコンの$\overline{MCLR}$ピンがGND電位となり、ハードウエアリセットがかかります。

⑨ピンショート防止ジャンパー(J5)

このジャンパーは、短絡時保護用抵抗を無効にするか、有効にするかを設定するものです。下図の通り、このジャンパーは220Ωの抵抗に並列に接続されており、ジャンパーソケットを外すと、タクトスイッチは220Ωの抵抗を介してVcc又はGNDと接続されます。



これによって例えば、マイコン側が意図せず出力設定になっていて、あるロジック状態になった際、誤ってタクトスイッチを押した時に大きな電流が流れずに220Ωの抵抗を介して電流が流れ、マイコンの破損を防止できます。

通常使用時は、J5のジャンパーは外しておくことを推奨します。

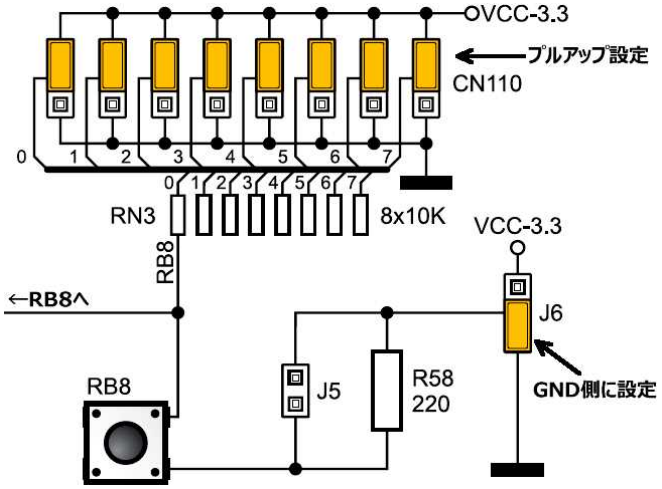
⑩アクティブLow/High選択用ジャンパ(J6)

⑫のタクトスイッチを押した時、該当のピンをLowにするのか、Highにするのかを設定するためのジャンパーです。

スイッチを押したときにLowになるものをアクティブLow、逆にスイッチを押したときにHighになるものをアクティブアクティブHighとよびます。J6のジャンパをVcc側にするとアクティブHighに、GND側に設定すると、アクティブLowとなります。

ただし、押しボタンスイッチの設定は、J6のアクティブLow/High設定と、⑬の各I/Oピンのプルアップ/プルダウンの設定を適切に組み合わせる必要があります。

例えば、J6をLow設定にした場合には、該当する押しボタンスイッチの接続されたポートとピンは、プルアップしておく必要があります。下記に例を示します。



上図の例では、定常時はRB8はHレベルですが、タクトスイッチを押すとLレベルになるよう設定した場合です。

⑪23個のI/Oピン状態表示LED群

各ビットの状態を表示できるLED群です。

該当のピンがHレベルになると点灯します。本製品で使用されているLEDの順方向電流は、1mAです。抵抗器を通してマイコンのピンと接続されています。

LEDのカソード側をGNDと接続するかどうかを、ディップスイッチSW4の1、2番スイッチによってポート単位で設定できます。

基板上にスイッチ横のシルク印刷に、どのポートのスイッチかが記載されています。

SW4の設定でLEDへの配線を接続しておくと該当ピンのHレベルの場合LED側に電流が流れます。よって、別の回路等と接続している場合、LEDに電流が流れることでHレベル時の電圧が低下することがあります。LEDを使用しない場合や、ピンに外部回路を接続する場合にはスイッチをOFFにしておくことが推奨されます。

⑫23個I/Oピン接続タクトスイッチ群

J6のジャンパー設定によって、アクティブHigh又はアクティブLowが選択可能な23個のタクトスイッチです。
タクトスイッチ群の左側にあるJ6をVccと刻印された側に設定すると、アクティブHighに、GNDと刻印された側に設定すると、アクティブLowになります。(詳しくは⑩の項目をご参照ください)

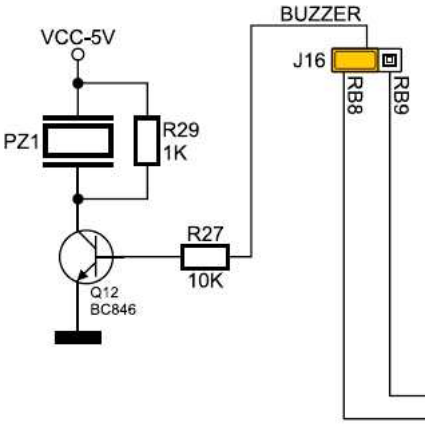
⑬全I/Oピン外部取り出しヘッダーピン及びプルアップ/ダウン設定

全I/Oピンをこの部分のヘッダピンから取り出せます。メスコネクタとオスコネクタと両方が用意されています。
ピンについてはボード上のシルク印刷で確認できます。

ピン単位でプルアップ/プルダウンがジャンパーソケットによって、設定できます。10KΩの抵抗器によって、Vcc(3.3V)へプルアップされます。プルダウンに設定すると、同様に、10KΩの抵抗器によってGNDへプルダウンされます。
外部の回路とI/Oピンを接続したい場合や、ピンの状態をオシロスコープ等の計測器で計測したい場合に使用します。
なお、全ピンではありませんが、⑱の端子台から取り出すことができます。

⑭圧電ブザー

実験用の圧電ブザーです。NPN型トランジスタによってマイコンの出力が増幅されて圧電ブザーを駆動します。
ジャンパーソケットJ16によって、トランジスタのベースをRB8又はRB9のどちらに接続するかを設定できます。
圧電ブザーは、そのままではスピーカーですので、電圧を印加しても音は聞こえません。主に20Hz～20KHzまでの可聴域の周波数内で発振波形を印加することで、音として聞こえます。



⑮静電容量式タッチセンサ(mTouch)用パッド

PICマイコンの開発元であるマイクロチップテクノロジー社では、静電容量変化を使ったタッチセンサシリーズの総称をmTouchと呼んでいます。(mTouchはマイクロチップ社の登録商標です。)

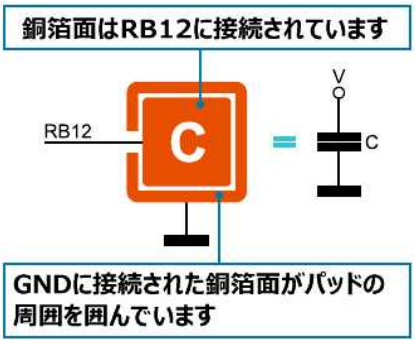
人間は、一定の水分や塩分を含み大地に接している巨大なコンデンサと考えることができます。冬場、金属に触れるなどして静電気でパチッと自らの体から金属へ放電が行われることからもそのことが分かります。その巨大なコンデンサである人間が、電子回路に接触すると様々な影響を与えることになります。この影響は通常、電子回路にとっては好ましくない場合が多いのですが、mTouchはこの影響をうまく利用して、タッチセンサーとして使う方法です。
実際には、プリント基板とGNDの間に寄生静電容量を形成し、指などの人体が近づいた時その静電容量が増大することを利用します。

このmTouchを使うにはコンパレーター(比較器)と、ラッチ回路が必要ですが、これらは対応PICの中に入っています。

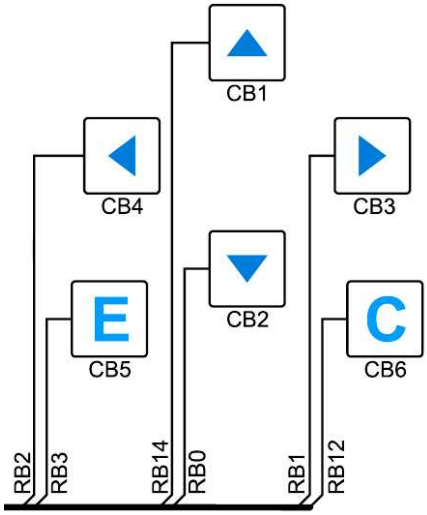
mTouchの具体的な使い方については本書では、記載していませんがCD-ROMにmTouchに関するマニュアルが収録されているので興味がある方はお読み下さい。ADコンバータを利用する方法と、一部のデバイスで搭載されているCTMUモジュールを使う方法があります。

mikroC PRO for dsPICで使えるサンプルプログラムが、CD-ROMに収録されていますので、こちらもぜひ併せてお試し下さい。

dsPICF-300では6つのパッドを搭載しており、中心部の銅箔部(基板内部の銅箔面)は各I/Oピンに、その周囲はGNDに接続されています。



パッド部はそのまま各ピンと接続されています。

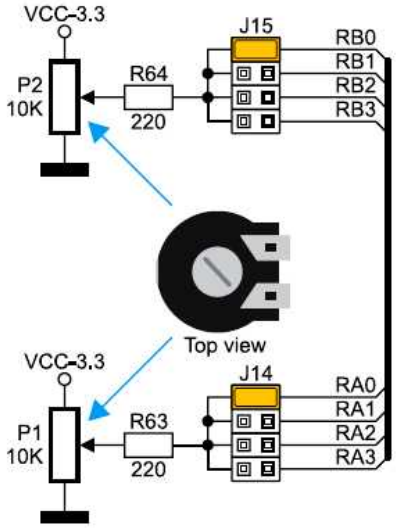


⑯ADコンバータ評価用ボリューム

ADコンバーター評価用として、P1及びP2の2つのボリュームが実装されています。ボリュームは10KΩで0V～3.3Vの電圧を各ボリュームの上側にあるJ14及びJ15のジャンパーによって設定したマイコンのピンに印加できます。

P1ボリュームはJ14で、P2ボリュームはJ15でそれぞれ設定します。

ADコンバータを使用しない場合にはジャンパーソケットは必ず外しておいてください。



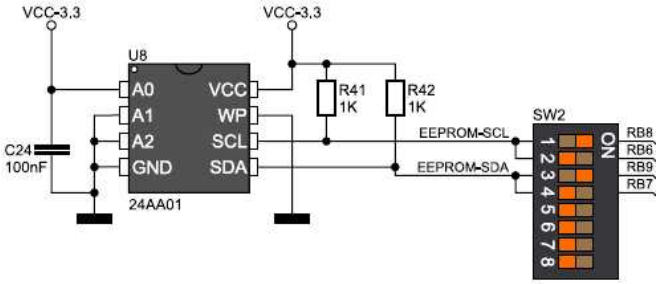
⑰実験用シリアルEEPROM、シリアルRAM

I2Cインターフェイス実験用に使用できるI2Cインターフェイス式の不揮発性メモリー(EEPROM)、24AA01(最大1kbitデータ)と、SPIインターフェイス実験用に使用できるSPIインターフェイス式のRAM、23K640(64kbit)が搭載されています。

EEPROMは、不揮発性メモリーで電源を切断しても記録内容が保持されますが、RAMは揮発性メモリーで電源断で内容は消去されます。但し、EEPROMよりRAMは高速に動作をするため、一時的なデータ格納用に使用されることが一般的です。

I2C通信式とSPI通信式両方のデバイスが搭載されていますので、各種同期式シリアルインターフェイスの実験や学習が可能です。

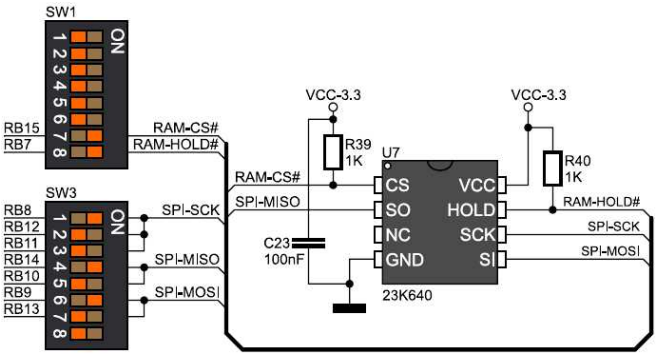
EEPROMの24AA01は、I2C通信で信号線のSDA及びSCLの結線は、ディップスイッチSW2の1～4番によって、マイコンのピンと接続するかを指定できます。SDAはRB9又はRB7と、SCLはRB8又はRB6と接続することができます。なお両信号線は、1KΩの抵抗器によって、プルアップされています。



I2Cのスレーブアドレスは A0～A2ピンの3ビットで決まり、配線がA0のみVccに接続されているため、"001"になっています。よって、スレーブアドレスは1になります。

RAMである23K640の各信号線は、ディップスイッチSW1とSW3にてSCK線やMISO線、MOSI線などの接続先を設定することができます。接続先については、基板のシルク印刷で確認できます。

なお、RAMのCS(チップセレクト)線と、HOLD線はそれぞれ1KΩの抵抗器でプルアップしてあります。



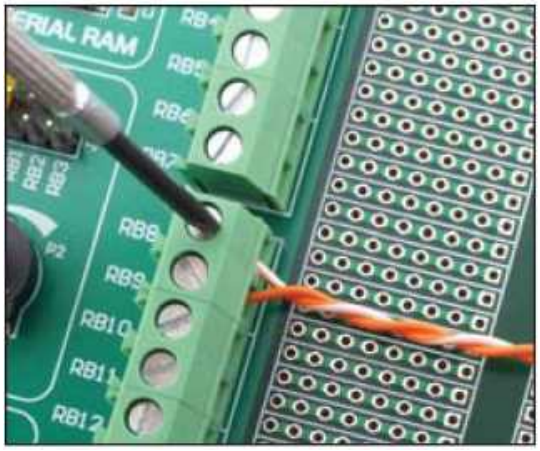
※それぞれのデバイスのデータシートは、CD-ROMに収録されております。

⑱実験用ピン取り出し端子台

本ボードにはユニバーサルエリアが設けられていますので、そこに自作回路等を作成した場合や、外部基板との接続などに利用できる端子台を搭載しています。

端子台は上部のねじを緩めて、側面の穴からリード線等を挿入してその後、ねじを締めて圧着することでケーブルを固定できます。

取り出せるI/Oピンについては、基板のシルク印刷を参照下さい。



⑲FT232RL搭載USB-UART変換ポート

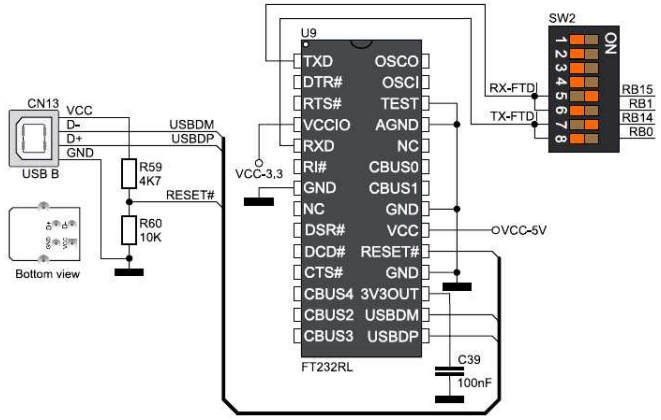
FT232RLは、FTDI社の有名なUSB信号とUART信号をブリッジするワンチップ変換ICです。

FT232RLのUSBポートをパソコンと接続すると、パソコンに仮想COMポートが作られます。(ドライバーソフトのインストールが必要です。)一方、FT232RLのUART側ピンは、通常のRS232CなどのUARTと同様、TX線、RX線といったピンが用意されています。これによって、ユーザーはUSBの専門知識を意識することなくUART信号を、USB端子経由でパソコンと送受信できます。パソコン側は仮想COMポートが作られていますので、RS232Cポートを開くのと同様、汎用的なシリアルターミナルソフトでポートをオープンすることで、通信が可能となります。

RS232Cポート搭載のパソコンが減る中、マイコンがパソコンと通信する場合は多く、そのインターフェイスはUSBに移行してきています。しかし、USBインターフェイスは複雑なプロトコルの中で動作しているため、プログラムもそれなりに複雑になり、USB通信の専門知識

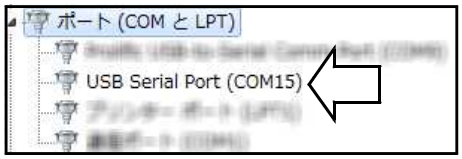
も必要となります。mikroCなどのCコンパイラでもUSB通信を行うための各種組込関数を搭載しており、PICマイコンから直接USB通信をすることもできますが、リソースがUSB通信に使われてしまい、他の処理が出来なくなるなどの問題もあります。FT232RLのようなワンチップUSB−UART変換ICを使うことで、従来のUART通信と同じ要領でUSB経由でのパソコンとの通信ができますので、様々な場面で使用されています。

本dsPICF−300ボードでは、マイコンとFT232RLとの接続はRX線とTX線の2線のみとなり、ハードウエアフロー制御には対応していません。RX線及びTX線については、ディップスイッチSW2の5〜8番スイッチで接続先を設定します。RX線はRB15又はRB1に、TX線はRB14又はRB0と接続することができます。マイコンのハードウエアUARTのピンとの関係を確認しながら設定をします。



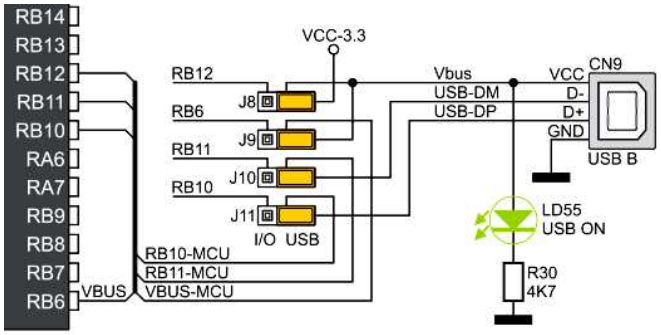
仮想COMポートドライバは、CD−ROM内の"FT232RLドライバー"フォルダに入っています。パソコンに接続した後、デバイスドライバーを要求されたら、このフォルダを指定してインストールしてご使用ください。なお、ドライバーインストールはチップ本体のドライバーと、仮想COMポートドライバーとで2回行われます。

新たに作成された仮想COMポートのポート番号は、デバイスマネージャで調べることができます。



⑳実験用USBポート

USB実験用のポートです。PICマイコンをUSBデバイスとして動作させる実験等に使用できます。USBの実験をする場合には、J8〜J11のジャンパーピンをUSB側にセットしてください。USBポートを使用しない場合には必ず、上記ジャンパーピンは、I/O側にセットしてください。設定を誤ると、マイコンやボード、場合によってはパソコン等を破損してしまう場合がありますので十分にご注意ください。

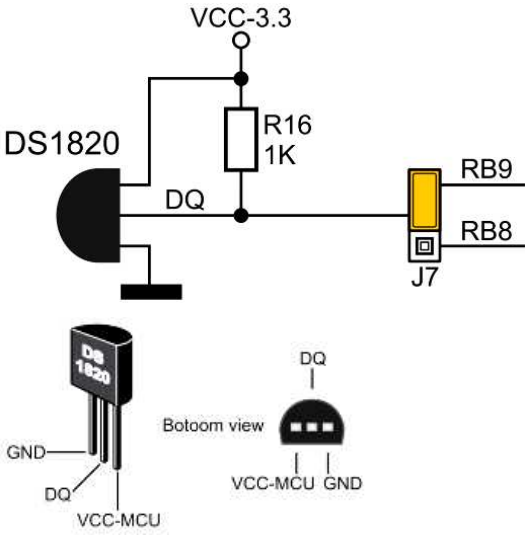


㉑温度センサIC(DS18S20)用コネクタ

ワンワイヤ通信方式を採用したダラスセミコンダクタ社(現マキシム社)のワンチップデジタル温度センサIC、DS18S20を接続するためのコネクタです。

DS18S20は非常にシンプルな構成で−55℃〜125℃までを計測できるワンチップ温度センサICです。当方でオプション品として販売しています。mikroCの組込関数から簡単に使用できます。

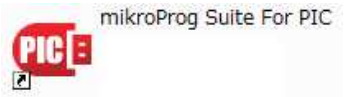
ジャンパーJ7によって、DS18S20のDQ線をマイコンのRB9又はRB8と接続するかを設定できます。DQピンは1KΩの抵抗でプルアップされています。使用する場合には、ソケットにDS18S20を極性を間違えないように装着し、ジャンパーソケットを適宜設定してご使用ください。



書き込みソフトウエアの使用方法

付属のUSB2.0マイコンライターの使い方を紹介します。
マイコンライターは、コンパイラで作成したHEXファイルをPICマイコンへ書き込む際に使用するソフトウエアです。
ソフトウエアは、mikroProg Suite for PICを使用します。付属のCコンパイラー、mikroCを使用すると、コンパイルと書き込みを連動させて行うことができて大変便利です。

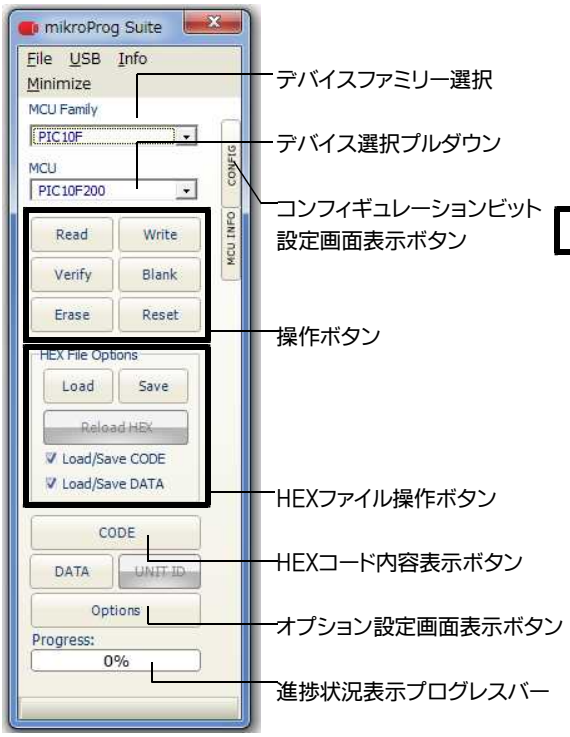
■PICプログラマーの起動



PICプログラマーは、Windowsのスタートメニュー又はデスクトップに作成されたショートカットから起動できます。

スタートメニューから起動する場合には、"スタート"→"プログラム"→"Mikroelektronika"→"mikroProg Suite For PIC"→"mikroProg Suite For PIC"の順でクリックして起動します。

■アプケーション概要



初期画面

"CONFIG"ボタンを押した時に右側に表示される画面

■操作の順番

次の順番でHEXファイルを書き込みます。

1 デバイス設定プルダウンより、書き込むデバイスを選択します。
最初に"デバイスファミリー選択"プルダウンからデバイスファミリーを選択します。"dsPIC33F"及び"PIC24シリーズ"が選択できます。

2 続いて"デバイス選択プルダウン"より書き込みを行うデバイスを選びます。

3 書き込むHEXファイルを読み込みます。
HEXファイル操作ボタンの"Load"ボタンを押します。
ダイアログが表示されますので、ファイルを選択します。
→ファイルが読み込まれるとステータスバーに、ファイル名が表示されます。

※同じファイルを再度ロードする場合には、いちいち上記のようにファイルを指定しなくても、"Reload HEX"ボタンを押すことで再読込ができます。

※ファイル名や、ファイルの配置してあるディレクトリ名に日本語や2バイト文字が含まれている場合、正しくHEXファイルの読み込みができません。必ず半角英数字になるようご注意ください。

4 一般的にはこれで書き込み準備は完了ですが、必要に応じてコンフィギュレーションレジスタの設定を確認します。
コンフィギュレーションレジスタは、プログラムを書き込む時にだけしか設定できないPICマイコン全体の動作や基本的な設定などを行う特殊なレジスタです。
コンフィギュレーションビットの設定が間違っているとプログラムや回路が正しくてもプログラムは動作しません。通常、コンフィギュレーションビットの設定内容は、プログラム内に埋め込まれているため、HEXファイルを読み込んだ時点で自動的に設定されます。
しかし、設定が正しくないとプログラムの動作に問題が発生することがありますので、書き込み前に確認されることをお奨めします。

"コンフィギュレーションビット設定画面表示ボタン"をクリックします。右側に設定ウインドウが展開します。

5 必要に応じてコード保護機能も設定します。
コードプロテクションを有効にすると、一度書き込んだHEXデータは読み出せなくなります。コードを不正にコピーされることを防止できます。通常は、"None"及び"Write protection Off"に設定します。

6 設定が完了、確認したら書き込みを行います。
dsPICF-300本体の"USB LINK"の黄LEDが点灯していることを確認してから操作ボタンの"Write"ボタンをクリックします。
書き込みを開始します。
書き込み動作中は、進捗表示プログレスバーに進捗状況が表示されます。完了するまで待ちます。

書き込みが完了すると、プログラムがすぐにボード上で動作を開始します。

■その他の機能

・Read機能
→PICマイコンからデータを読み込みます。
読み込んだデータは、"Code"ボタンで閲覧できます。またメニューバーの"File"→"Save HEX"で保存できます。

・Verify機能
→現在読み込まれているHEXファイルの内容と、PICに書き込まれているプログラムの内容が一致しているか検証します。

・Blankチェック機能
→現在装着されているデバイスのプログラムメモリがブランク(空)かチェックします。

・Erase機能
→現在装着されているデバイスのプログラムメモリの内容を消去します。

主な仕様

電源電圧:	USBバスパワー給電時 DC5V ACアダプタ給電時 DC9V
給電方法:	USBバスパワー又はACアダプタ
USB規格:	Ver.2.0対応
対応OS:	Windows98SE/ME/ Windows2000/XP(SP3)/VISTA/7 ※32ビット、64ビット両方に対応
対応デバイス:	dsPIC33F,PIC24シリーズ ※DIP形状のデバイスの限る
生産国:	セルビア/日本

対応デバイス一覧

dsPIC33FJ12GP201-DIP18, dsPIC33FJ12MC201-DIP20
dsPIC33FJ06GS102-DIP28, dsPIC33FJ06GS202-DIP28
dsPIC33FJ16GS402-DIP28, dsPIC33FJ16GS502-DIP28
dsPIC33FJ12GP202-DIP28, dsPIC33FJ12MC202-DIP28
dsPIC33FJ32GP202-DIP28, dsPIC33FJ128GP802-DIP28
dsPIC33FJ128GP202-DIP28, dSPIC33FJ64GP802-DIP28
dsPIC33FJ64GP202-DIP28, dsPIC33FJ32GP302-DIP28
dsPIC33FJ32MC202-DIP28, dsPIC33FJ128MC802-DIP28
dsPIC33FJ128MC202-DIP28, dsPIC33FJ64MC802-DIP28
dsPIC33FJ64MC202-DIP28, dsPIC33FJ32MC302-DIP28

PIC24F04KA200-DIP14, PIC24F04KA201-DIP20
PIC24F08KA101-DIP20, PIC24F16KA101-DIP20
PIC24F16KA102-DIP20, PIC24F08KA102-DIP20
PIC24FJ16GA002-DIP28, PIC24FJ32GA002-DIP28
PIC24FJ48GA002-DIP28, PIC24FJ64GA002-DIP28
PIC24FJ32GA102-DIP28, PIC24FJ64GA102-DIP28
PIC24FJ32GB002-DIP28, PIC24FJ64GB002-DIP28
PIC24HJ12GP201-DIP18, PIC24HJ12GP202-DIP28
PIC24HJ32GP202-DIP28, PIC24HJ128GP802-DIP28
PIC24HJ128GP202-DIP28, PIC24HJ64GP802-DIP28
PIC24HJ64GP202-DIP28, PIC24HJ32GP302-DIP28

サポート情報

当方では、dsPICF-300のサポートを行っております。以下のいずれかの方法でご質問をお寄せください。

■FAX番号　03-3700-3548
■電子メール　support@microtechnica.net

なお、他社製品に関することや自作回路に関するご質問にはお答え致しかねますのであらかじめご了承ください。

ソフトウエアはアップグレードされることがあります。アップグレードされると対応デバイスが増えたりバグが修正されたりします。
アップグレードの情報などは弊社のwebページにてお知らせ致しますので、弊社webページを定期的にご確認ください。
http://www.microtechnica.net/

PICマイコン用の統合開発環境,MPLABはマイクロチップ社製のソフトウエアです。バージョンアップ等の情報についてはマイクロチップ社のホームページより告知されますので,下記ページを定期的にご確認ください。
http://www.microchip.com/ （英語サイト）

マイクロテクニカ
〒158-0094　東京都世田谷区玉川1-3-10
TEL: 03-3700-3535　FAX: 03-3700-3548
(C)2011 Microtechnica All rights reserved

C言語とは

C言語は、ANSIという団体によって規格化された高級言語です。高級言語とはアセンブラ言語のように機械語に近いレベルの開発言語ではなく、より人間がわかりやすい言語体系の開発言語を指します。C言語を導入することでより簡単に、合理的にPICマイコンの開発ができるようになります。

C言語では様々なデータ型を扱えるほか、データを指し示すポインタや配列、関数などが使用でき、より高度なプログラムを記述することができます。

本dsPICF-300に付属のmikroCコンパイラーはANSI規格に準拠したPICマイコン用のC言語です。本製品に付属のCD-ROMに収録のmikroCはdsPIC30F及びdsPIC33Fシリーズをはじめ、PIC24にも対応しています。体験版のため開発できるプログラムのコードサイズが6KBまでに限定されていること以外、すべての機能が使用できます。mikroCが気に入れば、コード制限のないフル機能版を入手することができます。

mikroCは、一般的なC言語としての機能の他にPICマイコンの機能をフルに使用できるよう様々な組み込み関数を搭載しています。組み込み関数を使用することで、LCD制御やADコンバーターの操作、UARTなど様々な機能を簡単に実現できます。PICマイコンの開発に便利な様々な組み込み関数が搭載されています。

チュートリアル　～dsPICF-300をmikroCで使おう～

早速付属のmikroCを使ってC言語でプログラムを作り、実際にdsPICF-300のボード上で動作させてみましょう。本チュートリアルを一通り試すことで、dsPICF-300の使い方及びmikroCの使い方を体験・習得できます。

なお本マニュアルではC言語の基本から解説することはできませんので、C言語の基本については専門書籍をご参照ください。

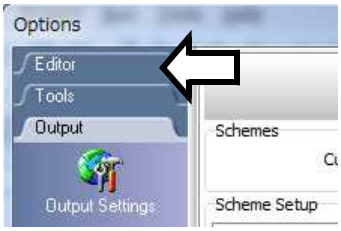
■開発環境の設定

では最初にmikroC Cコンパイラーの開発環境を設定しましょう。

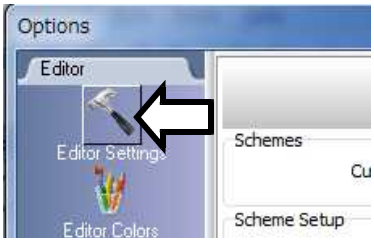
1 mikroCを起動します。
"スタート"→"プログラム"→"Mikroelektronika"→"mikroC PRO for dsPIC"→"mikroC_dsPIC"の順でクリックして起動します。

2 起動すると、開発画面が表示されます。
初期設定では、文字が小さく見にくいのでフォントサイズを大きく変更しましょう。

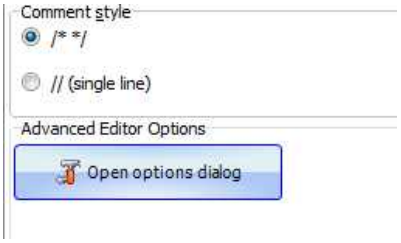
3 メニューバーから"Tools"→"Options"をクリックします。ダイアログが表示されますので、左側のタブから"Editor"をクリックします。



4 続いて、"Editor Settings"のボタンを押します。



5 "Open Option dialog"ボタンをクリックします。



6 "Editor Options"ダイアログが表示されますので、画面右下の"Editor Font"部分にある"Font"ボタンをクリックすると、フォントを選択できるダイアログが表示されますので、サイズを調節してください。

※実用的な大きさとしては、画面の解像度にもよりますが、14ポイント程度が見やすいです。
※右上にある"Font"ボタンではありませんので注意してください。
※フォントの種類は変更しないでください。

設定したら"OK"ボタンを押して完了します。さらにOptionsの画面では、"Apply"ボタンをクリックして設定内容を反映させた後、"OK"ボタンを押して完了します。

※その他にも詳細な設定が可能ですが、トラブルを避けるためフォントの変更以外は行わないことをお奨めします。

チュートリアル①　～PORTBの制御とボタン処理～

では早速プログラムを作って開発を体験してみましょう。最初のチュートリアルでは、まず全体の体験ということで、簡単なプログラムを作ります。

16ビットの変数を用意して、そこに入れた値を2進数でPORTBに出力するプログラムを作ってみましょう。変数を2つ用意して、外部のスイッチ入力によって、どちらの変数の値をRBに出力するのか指定できるようにしてみます。

デバイスは標準で付属しているPIC24F16KA102を使用することにしましょう。

ここでPIC24F16KA102についてその使用を簡単にご紹介しておきましょう。

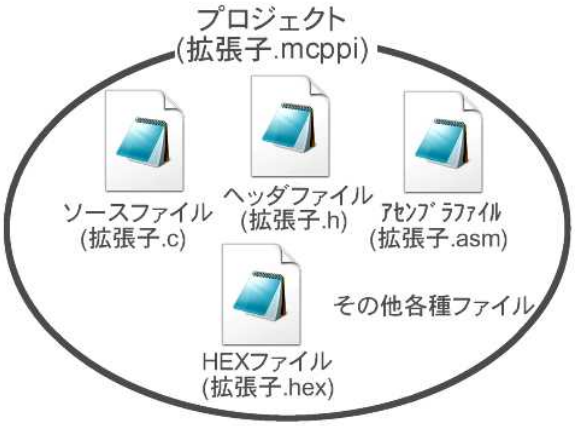
CPUスピード	16MIPS　(@32MHz)
プログラムメモリ	16kバイト
RAM	1536バイト
動作周波数	最大32MHzまで
動作電源電圧	1.8V～3.6Vまで
内蔵発振子	8MHz
逓倍回路(PLL)	×4倍（8MHzで最大32MHz）

内蔵発振子に8MHzが選択でき、PLLと組み合わせると内蔵発振子だけで32MHzの動作が実現できる高性能なPICマイコンです。

【1】プロジェクトの新規作成

プログラムを作り始める前に、プロジェクトを新規に作成します。mikroCでは1つのファームウエアに対して1つのプロジェクトとして、プロジェクト単位で管理します。

プロジェクトには、これから作成するC言語のソースプログラム(拡張子.c)や、場合によって使用することとなるヘッダーファイル(拡張子.h)などのファイルが登録されます。その他、コンパイル後のHEXファイルなどもプロジェクトに含まれる1ファイルとして管理されます。



よくある間違いとして、プロジェクトの意味を理解していないと画面上に表示されているソースプログラムをコンパイルしているつもりなのに、実際には別のソースプログラムがコンパイルされていて思ったような動作をしなかったり混乱してしまったりというトラブルに繋がります。新しいプログラムを作るときには、必ずプロジェクトを新規に作成するようにしてください。

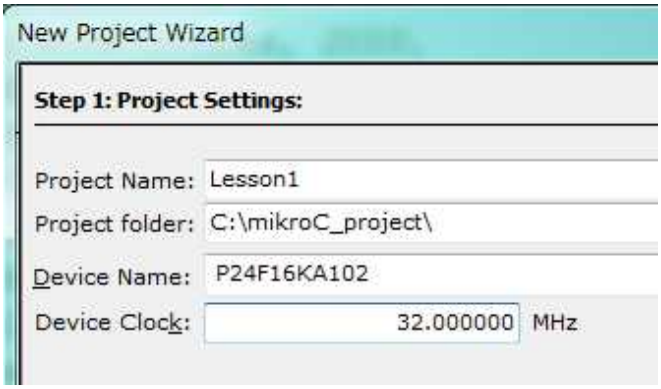
1 新規にプロジェクトを作成してプログラムを作りましょう。
メニューバーから"Project"→"New Project"をクリックします。
mikroCではウィザード形式でプロジェクトを新規に作成します。ウィザード画面が表示されたら、"Next"ボタンを押して続行します。

2 デバイスを選択します。プルダウンから"P24F16KA102"を選択します。
クロック周波数を設定します。今回は、dsPICF-300に最初から付いている水晶発振子8MHzを使用しますが、PIC24F16KA102に内臓のPLLにて4倍して、最高速度の32MHzで動作させてみましょう。
"Device Clock"の部分に、"32" と入力して32MHzに設定します。
小数点以下は自動的に0が挿入されますので、単に 32 とだけ入力してください。

3 プロジェクトの保存場所と、プロジェクト名を指定します。
"Project Name"の部分には適当なプロジェクト名を付けます。ここでは、"Lesson1"と付けてみます。なお、プロジェクト名はどんな名前でもいいですが、日本語や2バイト文字(全角文字)は使用できません。
その下の"Project Folder"の部分には、プロジェクトファイルを保存するディレクトリを指定します。ここでは、例としてCドライブの直下に"mikroC_project"というフォルダを作ります。

⚠ ディレクトリ名及びファイル名には、日本語や全角文字などの2バイト文字は使用できません。必ず半角英数字で指定できるディレクトリ及びファイル名としてください。

すべて設定すると、次のような内容になります。
内容を確認して"Next"ボタンをクリックします。



※¥マークは、英語環境なのでバックスラッシュで表示されます。

4 続いて "Step2: Select files you want to add to project" のウィザードでは何も変更等はせずに"Next"ボタンを押します。

5 "Step3: Select initial state for library manager"のウィザードでは、"Include None (Advanced)"にチェックを入れいください。
"Next"をクリックします。

※ここでのチェックは、mikroCの組込関数をプロジェクトに含めるかどうかを選択するものです。すべて含めると、すべての組込関数が使えますが、プログラムのサイズが大きくなるほか、ピン数の少ないデバイスではエラーがでることがあります。ライブラリは、プログラムを作りながら、1つずつ必要な物を手動でインクルードできるので、ここでは含めない設定とします。

6 最後に"Finish"ボタンをクリックして完了します。

この状態でプログラムのひな形ができあがりました。mikroCの優れている点は、このウィザードによって、使用するデバイスに関連するファイルが自動的に読み込まれ、動作周波数も設定できることです。デバイス定義ファイルなどは別途読み込む必要がありません。

【2】プログラムの記述

ではプロジェクトが作られましたので、プログラムを記述しましょう。mikroCでは、新規にプロジェクトを作成すると、"プロジェクト名.c"という拡張子がcのファイルが自動的に作られます。基本的にはこのファイルにC言語のプログラムを記述していくことになります。

新規にcファイルが作られると、main関数だけが記述された状態になります。C言語の基本的な解説は省略しますが、C言語ではすべてのプログラムは、main関数から始まります。関数は1つのプログラムの中にmain関数1つだけでもよいですし、適宜ユーザーが追加して作ってもかまいません。

voidというのは、型の名前でmain関数はvoid型であることを意味しています。void型とは、、引数や戻り値が存在しないことを意味する型です。

では早速プログラムを記述してみましょう。C言語では大文字・小文字は基本的には関係ありません。またスペースや空白の行も基本的には関係ありません。但し、統一した方が見やすいので、一般的には関数や変数は小文字で、レジスタ名は大文字で記述することが多いです。

プログラムは見やすいようにタブやスペース、改行を適宜入れながら記述していきます。

では次のようにプログラムを入力してみましょう。

```
void main(){

  unsigned int a , b ;
  AD1PCFG = 0x1C3F;
  TRISB = 0 ;
  TRISA = 0x3;
  PORTB = 0 ;
  LATA = 0;
  LATB = 0;
  a = 5200 ;
  b = 30000 ;

  while(1){
     if (PORTA == 0x1) PORTB = a ;
     if (PORTA == 0x2) PORTB = b ;
  }
}
```

【3】コンフィギュレーションビットの設定

PICには、コンフィギュレーションビットという特殊なレジスタがあります。これは、PICの動作に関わる最も基本的な設定を行うもので、プログラムをPICに書き込む時に設定するものです。

代表的なものには、例えば発振子の種類の指定があります。発振子は大きく分けて外部発振子か、内蔵発振子かに別れ、さらに外部発振子の場合にはその周波数によって設定が分かれます。その他、ウォッチドッグタイマーのON/OFFの設定などを行うのがコンフィギュレーションビットです。

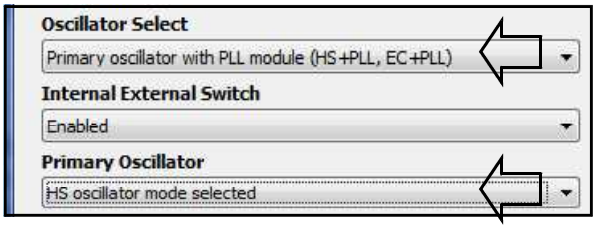
mikroCでは、コンフィギュレーションビットの設定は、生成するHEXファイルに埋め込んだ形でコンパイルを行います。HEXファイルにコンフィギュレーションビットの設定を埋め込んでおけば、書き込みの時にその都度、設定をしなくても書き込みソフトウエアはHEXファイルを読み込むと自動的にコンフィギュレーションビットの値をセットするので、手間が省け、間違いも少なくなります。次の手順でコンフィギュレーションビットの値を設定しましょう。

1 メニューバーから"Project"→"Edit Project"を選択します。

2 ダイアログが表示されます。ここで1つずつ設定を行います。
PIC24シリーズは高機能なデバイスのためコンフィギュレーションビットの設定内容もかなり複雑になっています。コンフィギュレーションビットの設定内容が間違うとプログラムが正しくてもプログラムが動作しなかったり、予想しない動作をしたりと不具合の原因となりますので、間違いがないよう正しく設定をする必要があります。しかしここで変更するのは2つの項目だけです。

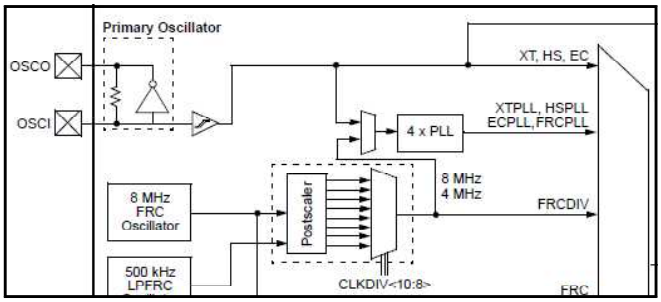
最初に最も基本的な設定内容を読み込みますので、ダイアログ右下の"Default"ボタンを押します。この状態から設定を行います。
変更するのは下記の項目です。

・Oscillator Select
→Primary oscillator with PLL module (HS+PLL,EC+PLL)
・Primary Oscillator
→HS oscillator mode selected



その他の設定はそのままにしておきます。設定が完了したら"OK"ボタンをクリックします。

設定した内容ですが、今回は発振子についての設定を変更しただけです。発振子は、"Oscillator"と書かれた項目です。発振子はマイコンの外部に取り付けられた8MHzの水晶発振子を使いますので設定は"HS"になります。さらにその8MHzのクロックをPIC内部でPLL回路を使って4倍に逓倍して8MHz×4倍＝32MHzとしますのでその設定をしました。どうしてそのような設定になるのかは、データシートのオシレータの内部ブロック図を見るとよく分かります。データシート91ページには内部のブロック図が記載されています。



上図は抜粋ですが、OSC0とOSC1には現在8MHzの水晶発振子が取り付けられていますので、Primary Oscillatorは5MHz以上の発振子が取り付けられていますので、"HS"モードになります。これが4MHz以下の場合には、"XT"モードにします。
続いて、その先ですがその水晶発振子のクロックをそのまま使う場合には、"XT, HS, EC"の選択でいいですが、今回は4逓倍しますので、4×PLLという方に流れが変わります。すると、4×PLLの出力は"XTPLL,HSPLL・・・"という部分になりますので、コンフィギュレーションビットの設定値でOscillator Selectの部分を「Primary oscillator with PLL module (HS+PLL,EC+PLL)」に設定したというわけです。複雑に見えるクロック回路ですが、ブロック図を追っていくと簡単に設定値を決めることができます。

その他設定項目はありますが、ここではその他の項目は今回のプログラムでは関係ないので、デフォルトのままにしておきます。より詳細にコンフィギュレーションビットの設定内容を知りたい場合には、PICマイコンのデータシートをお読みください。

なお、以後すべてのチュートリアルで本設定を基本としますので、この設定内容を覚えておいてください。

【4】ボードの設定とコンパイル及び書き込みの実行

C言語で書かれたプログラム(ソースプログラムといいます)を、PICに書き込める機械語のHEXファイルにする作業をコンパイルと呼びます。mikroCとdsPICF-300の組み合わせの場合、コンパイルとコンパイル後に生成されたHEXファイルの書き込みを連動して行うことができます。コンパイルでエラーが発生した場合には、書き込みは実行されません。

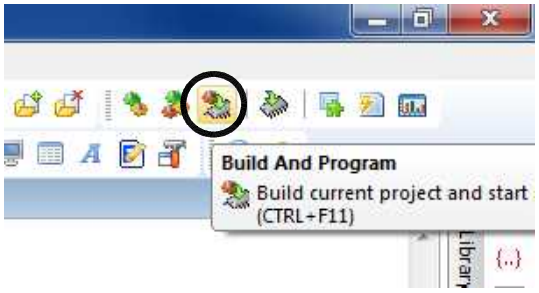
ここでは最初にボードの設定を行った後、コンパイル→書き込みを実行してみましょう。書き込み後はすぐにプログラムが動きますので、書き込みを実行する前に、dsPICF-300のボードの設定をしておく必要があります。

1 dsPICF−300ボードの各種設定を行いましょう。今回はRBはLEDに接続し、RAは定常時プルダウンに設定し、タクトスイッチが押された時にVcc(+3.3V)に接続されるようアクティブHigh設定にします。次のように設定してください。

・ディップスイッチSW4
→"PORTB"(2番)のみスイッチをON側にします。
その他のスイッチはすべてOFF側にセットします。
・PORTA部のジャンパー設定
→"Pull−Down"間(下側)をジャンパーソケットでショートします。
・ジャンパーJ6(タクトスイッチ設定)
→"VCC3"側をジャンパーソケットでショートします。

上記のように設定すると、PICのPORTBのピンはLEDに接続され、該当のビットがHレベルになるとLEDが点灯します。
また、PORTAは常時Lレベルで、タクトスイッチが押されると、該当ビットがHレベル(+3.3V)になります。

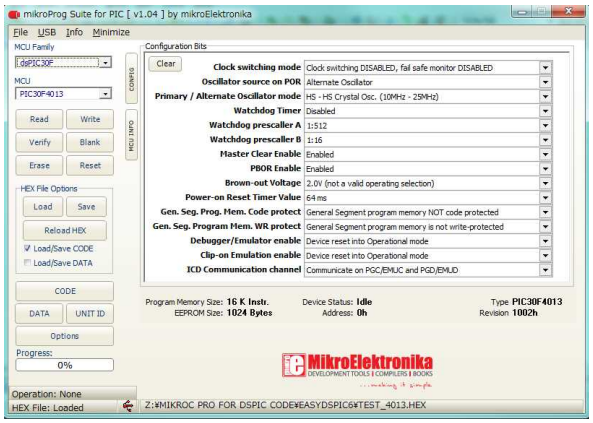
2 設定ができたら、コンパイルと書き込みボタンをクリックします。



ツールバーの"Build and Program"ボタンを1回クリックします。

3 クリックするとコンパイルが始まります。
ウインドウ下部のメッセージウインドウにコンパイル結果が表示されます。エラーがなければ、"Finished successfully"と緑の文字で表示されます。
エラーが表示された場合には、プログラム内容に間違いがないか確認してください。
生成されたHEXファイルは、プロジェクトを保存したフォルダ内に生成されています。

4 コンパイルが正常に完了すると自動的に書き込みソフトウエアが起動し、書き込みを実行します。進捗状況がプログレスバーで右下に表示されます。



"Configuration Bits"と書かれたプルダウンのまとまり部分が先に設定したコンフィギュレーションビットの設定です。

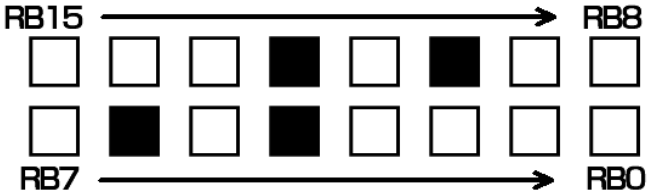
※mikro Prog Suiteの画面表示が小さい(コンフィギュレーションビットの設定ウインドウが表示されない)場合には、mikro Prog Suiteの"Config"ボタンをクリックすると、画面が展開します。

【5】動作確認

書き込みが完了するとプログラムがdsPICF−300ボード上ですぐに動きます。動作を確認してみましょう。

dsPICF−300のタクトスイッチ群のなかにあるRA0のスイッチを押します。RBに接続されたLEDが点灯します。RA0なので変数aの値、ここでは5200が2進数でLEDに現れます。

5200は2進数表記では　0001010001010000　となります。左側が上位ビット、右側が下位ビットです。RB15が上位ビット、RB0が下位ビットになりますので、下記のように点灯しているはずです。



※□＝消灯、■＝点灯

同様にして、RA1のスイッチを押すと変数bの値、すなわち30000が2進数としてRBのLEDに表示されます。
(30000 = 0111010100110000)

変数の値を色々と変えたり、スイッチの種類を増やしたりして実験を行ってみましょう。思い通りにプログラムが動作することを確認してください。

【6】プログラムの解説

void main(){…}

main関数です。C言語では1つのプログラム内に必ず1つのmain関数が必要です。voidはmain関数の型で戻り値がない関数であることを宣言しています。{ }で囲まれた範囲がmain関数の範囲となります。

unsigned int a , b ;

符号なしのint型変数を宣言しています。C言語では文の最後には必ず ; (セミコロン)をつけます。
符号なしのint型の場合扱える値は、0〜65535までです。

AD1PCFG = 0x1C3F;

AD1APCFGというのは、レジスタの名前でADコンバータの入力ピンなどについて制御を行うレジスタです。PICマイコンは電源投入時や、リセットをした後のデフォルト状態ではADコンバータ入力に割り当てられているピンはアナログ電圧の入力ピンとして設定されています。この状態ですと、通常のデジタルI/Oピンとしてプログラム上か

らHレベルにしたり、Lレベルにしたり、又は入力ピンとしてHレベルなのか、Lレベルなのかという状態を取得できません。
そこでこのレジスタを設定したすべてのピンについて、デジタルI/Oピンに設定しているのがこの行です。
mikroCでは、このように変数に値を代入するのと同じようにして、任意のレジスタに代入演算子(=)を用いて値を代入できます。

TRISB = 0 ;
TRISA = 0x3;
TRISレジスタは、ピンの入出力方向を設定するレジスタです。
該当のビットが0の場合は出力、1の場合は入力となります。本チュートリアルではRBは全ピン出力、RAはRA0とRA1が入力なので0x3として設定しています。
C言語では16進数の値を表現する時は数値の前に0xを付けます。

※2進数の時は"0b"(ゼロビー)を、10進数の時には数字をそのまま記述します。

PORTB = 0 ;
PORTBに0を代入しています。 =(イコール1つ)は代入演算子で右辺の値を左辺に代入します。
PORTBの出力を全ビットLレベルにしています。

a = 5200 ;
b = 30000 ;
int型変数に値を代入しています。
代入は、右式から左式に代入されます。代入には代入演算子(=)を用います。

while(1){…}
while文は( )内の値が1又は真であれば{ }内の処理を繰り返す繰り返し文です。ここでは評価式を1としていますので永久に{ }内が実行されます。
C言語では処理を永久ループさせたい場合にはよくwhile文を使用します。

if (PORTA == 0x1) PORTB = a ;
if文を使用した入力判定部分です。
( )内の評価式が真の時にその後ろに記述された文が実行されます。
mikroCでは、レジスタの値を直接評価式の中で判定することができます。
==(イコール2つ)は比較演算子の「等しい時」を表す演算子です。
よくある間違いとして=を1つしか記述しない場合があります。=が1つの場合には代入演算子となってしまい正しく動作しません。文法上は間違いではないのでエラーがでないため間違いを見逃してしまいプログラムが思うように動かない・・・ということがあります。

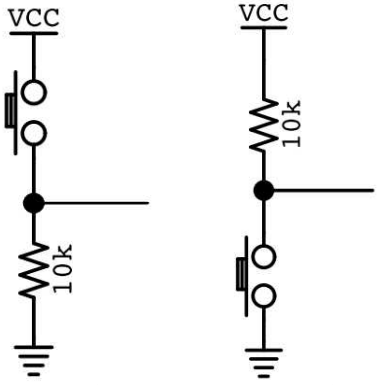
PORTAは、スイッチが押されるとそのビットがHになります。RA0のタクトスイッチが押されると、0000 0000 0000 0001 となり、PORTAの値は1となります。同様にして、RA1のタクトスイッチが押されると、0000 0000 0000 0010 となりますので、PORTAの値は2となります。これをif文で判定して処理を分岐しているわけです。

PORTB = aで、変数aの値をそのままPORTBレジスタに代入しています。

●スイッチ入力の論理について●
デジタルI/Oピンへのスイッチ入力には2つの状態遷移があります。

1つ目は、スイッチを押していない時にI/OピンがHレベルでスイッチを押した時にLレベルにする「アクティブLow」という方法。
2つ目は、スイッチを押していない時にI/OピンがLレベルでスイッチを押した時にHレベルにする「アクティブHigh」という方法。

チュートリアル1では、スイッチを押した時該当のピンがHレベルになるので、アクティブHighの設定でした。回路図にすると下図のAのようになります。



図A:アクティブHigh　図B:アクティブLow

図Aでは、常時I/Oピンは10kΩの抵抗器でプルダウンされています。電圧は印加されていないので、抵抗による電圧降下はありませんので抵抗器の両端の電圧は同電位(GND電位)です。よって論理はLです。スイッチが押されるとVccと接続され、VccからGNDに向かって10kΩの抵抗を介して電流が流れます。抵抗器の両端の電圧は抵抗器による電圧降下が発生し、Vccと等しくなりますので論理がHになります。

図Bはアクティブlowの場合です。常時I/Oピンは10kΩの抵抗器でプルアップされています。この時I/OピンはほぼVcc電位なのでHレベルです。スイッチを押すと、GNDと繋がりますのでLレベルとなります。(スイッチは0Ωですのでスイッチの両端に電圧は発生しませんので、V=I×0Ωで電圧降下は0Vです)

このようにスイッチ入力の状態を作る場合には、アクティブHかLどちらにするかを決めて、それに従って回路を作ります。dsPICF-300のボードでは各ピンのプルアップ/ダウンについてはジャンパーピンで設定できますし、スイッチを押した時の接続ピンをVccにするかGNDにするかはJ6のジャンパーで設定できますので、この2つのジャンパーの組み合わせでスイッチの論理状態を作ります。

チュートリアル②　～LCDの使用と、配列・ポインタ～

■プログラムの作成

dsPICF-300に装着されている2行16文字のLCDに文字列を表示します。C言語では、文字列を扱う場合配列を使用します。

固定した文字列をLCDに表示させてみましょう。

プログラムの作り方は、チュートリアル①と同じです。新規にプロジェクトを作成し、新しいプログラムとして開発します。"Edit Project"からコンフィギュレーションビットの値を設定するのを忘れないようにしてください。

```
sbit LCD_RS at LATB4_bit;
sbit LCD_EN at LATB5_bit;
sbit LCD_D4 at LATB6_bit;
sbit LCD_D5 at LATA0_bit;
sbit LCD_D6 at LATA1_bit;
sbit LCD_D7 at LATA4_bit;

sbit LCD_RS_Direction at TRISB4_bit;
sbit LCD_EN_Direction at TRISB5_bit;
sbit LCD_D4_Direction at TRISB6_bit;
sbit LCD_D5_Direction at TRISA0_bit;
sbit LCD_D6_Direction at TRISA1_bit;
sbit LCD_D7_Direction at TRISA4_bit;

void main(){
     char txt1[] = "HELLO";
     char txt2[] = "WORLD";

     ADPCFG = 0xFFFF;

     Lcd_Init();

     Lcd_Cmd(_LCD_CLEAR);
     Lcd_Cmd(_LCD_CURSOR_OFF);
     Lcd_Out(1,1,txt1);
     Lcd_Out(2,1,txt2);
}
```

プログラムを記述した後はチュートリアル①と同様の手順でコンパイルをして、HEXファイルを生成後、書き込みを実行します。ボードの設定は特に注意する点はありませんが、ディップスイッチはすべてOFFにしておいたほうがよいでしょう。(SW3の8番だけはLCDのバックライトのスイッチですのでONにしてください。)

LCDの1行目に"HELLO"と表示され、2行目に"WORLD"と表示されれば成功です。

※ディップスイッチとジャンパーソケット

dsPICF-300にはdsPICマイコンと各種周辺回路を物理的に接続するかどうかを設定する多数のディップスイッチとジャンパーソケットがあります。効率よく、かつ他の回路への影響を最低限にしながら各種実験ができるようになっているためです。

しかし場合によっては、思いがけずスイッチがONになっていたり、ジャンパーソケットによってジャンパーされていると、そちらの回路側に電流が回り込み、流れる電流が多いと電圧降下が発生して、思い通りの動作が得られない場合があります。使用しない回路からは、切り離しておくようにすると、思いがけないトラブルを避けることができます。

■プログラムの解説

sbit LCD_RS at LATB4_bit; ～
sbit LCD_D7_Direction at TRISA4_bit;

main関数の前にsbitタイプを用いてLCDの物理的な配線状態を定義します。

sbitタイプは、PICマイコンのSFR(レジスター)の特定のビットに対してアクセスするため機能を提供します。(#defineのようなプリプロセッサの働きがあります。)

ここでは、その1つの使い方としてLCDの接続状態をあらかじめ定義しています。sbitは、TRISレジスタに対しても特別な書式として記述することができます。LCD関連のライブラリを使用する場合には、main関数の前で、この定義を記述してください。

この配線記述は、dsPICF-300のボード配線にあわせて記述しています。

※この定義をmain関数の中で記述してしまうとエラーになります。

char txt1[] = "HELLO";

char型配列を宣言しています。char型は文字を扱う8ビットの型です。C言語では文字はASCIIコードにて扱われます。

通常配列を宣言する際には要素数(インデックス値)を[　]内に入力しますが、[　]内を空欄にすると自動的に要素数が設定されます。

配列や変数を宣言する際に初期値を代入できますが、char型に限り文字列で初期化できます。"(ダブルクォーテーション)で囲むと文字列として認識されます。

Lcd_Init() ;

LCDを初期化する関数です。LCDを使用する前に記述します。

Lcd_Cmd(_LCD_CLEAR);

Lcd_Cmd(　)関数は、LCDの制御コマンドを送信する関数です。

ここでは、画面をクリアーするコマンドと、カーソル(描画位置に点滅するカーソル)を表示しない設定にしています。

※mikroCのマニュアルは、"Help"→"Help"で見ることができます。

Lcd_Out(1,1,txt1) ;

Lcd_Outは4ビットモードでLCDに文字や文字列を表示する時に使用します。書式は下記の通りです。

Lcd_(行, 列, char *text);

*textは、char型のポインタという意味です。ここでは、配列名を記述することで配列の0番目の要素のアドレス値を指定しています。

*textは、char型のポインタという意味です。ここでは、配列名を記述することで配列の0番目の要素のアドレス値を指定しています。

ポインタについての詳細はC言語の専門書籍を参照ください。

※typedefを使ってみる

次に紹介するのは、typedefという命令を使った方法です。LCDへの表示内容は先ほどのプログラムと同じですが、typedefを使ったポインタ、配列の使い方を紹介します。

typedefは、定義済みの型に別の名前をあらためて定義するときに利用する方法です。例えば「char型」はすでにC言語では定義済みの型ですが、typedefを使うとchar型という型名を別の型名として定義することができます。

typedef char moji;

例えば上記のように記述すると、char型という型名は以後mojiという新しい型名として使えるようになります。

次のプログラムを作ってみましょう。

```
sbit LCD_RS at LATB4_bit;
sbit LCD_EN at LATB5_bit;
sbit LCD_D4 at LATB6_bit;
sbit LCD_D5 at LATA0_bit;
sbit LCD_D6 at LATA1_bit;
sbit LCD_D7 at LATA4_bit;

sbit LCD_RS_Direction at TRISB4_bit;
sbit LCD_EN_Direction at TRISB5_bit;
sbit LCD_D4_Direction at TRISB6_bit;
sbit LCD_D5_Direction at TRISA0_bit;
sbit LCD_D6_Direction at TRISA1_bit;
sbit LCD_D7_Direction at TRISA4_bit;

void main(){
     typedef char *moji, Array[5];

     moji txt1 = "Hello";
     moji txt2 = "World";

     AD1PCFG = 0x1C3F;
     Lcd_Init();
     Lcd_Cmd(_LCD_CLEAR);
     Lcd_Cmd(_LCD_CURSOR_OFF);

     Lcd_Out(1,1,txt1);
     Lcd_Out(2,1,txt2);
}
```

ポイントは、typedef char *moji, Array[5]; です。この1行で、char型で要素が5つの配列を"moji"という別の新しい型名として定義しています。ここで新しい型を定義したら、後は通常の変数宣言のように宣言するだけで、新しい型が使えます。

typedefはプログラマがプログラムの内容を理解しやすいようにするための方法で、例えば上のプログラムのように配列の要素数が決まっている配列をいくつも作りたい場合には宣言が楽なるというメリットもあります。

またtypedefは構造体を定義する時にもよく使われます。

チュートリアル③　～非同期式シリアル通信とADコンバーター～

■プログラムの作成

非同期式シリアル通信(UART)を使用して、RS232C通信をするプログラムを作ります。

dsPICF-300ではUART信号をRS232C信号にする回路と、同じ信号をUSBの仮想COMポートに変換する回路が2つ搭載されています。RS232Cは最近のパソコンにはポートが付いていないことが多いため、USBポートでRS232Cの信号をエミュレートするUSB-UART変換回路を搭載しました。このチュートリアルでは、このUSB-UART変換回路の方を利用してパソコンとUART通信をするプログラムを作ります。

RS232C経由で文字'A'(0x41)を受信すると、ADコンバータのチャンネル4(AN4 , RB2)の値を文字列として出力します(10ビット分解能)。同様にして、文字'B'(0x42)を受信すると、LCDにADコンバータで取得した値(AN4)を数字で表示します。

なおUARTの通信速度は9600bps、データ長8ビット、1ストップビット、ノンパリティの設定とします。

```
sbit LCD_RS at LATB4_bit;
sbit LCD_EN at LATB5_bit;
sbit LCD_D4 at LATB6_bit;
sbit LCD_D5 at LATA0_bit;
sbit LCD_D6 at LATA1_bit;
sbit LCD_D7 at LATA4_bit;

sbit LCD_RS_Direction at TRISB4_bit;
sbit LCD_EN_Direction at TRISB5_bit;
sbit LCD_D4_Direction at TRISB6_bit;
sbit LCD_D5_Direction at TRISA0_bit;
sbit LCD_D6_Direction at TRISA1_bit;
sbit LCD_D7_Direction at TRISA4_bit;

void init(){
      ADPCFG = 0xFFFF;
      Lcd_Init();
      Lcd_Cmd(_LCD_CLEAR);
      Lcd_Cmd(_LCD_CURSOR_OFF);
      UART2_Init(9600);
      Delay_ms(100);
}

void main(){

  char uart_rd, txt[7];
  unsigned int AD_res;

  init();

  while (1) {
    AD_res = ADC1_Read(4);
    intTostr(AD_res,txt);

    if (UART2_Data_Ready()) {
      uart_rd = UART2_Read();
      switch(uart_rd){
       case 0x41: UART2_Write_Text(txt);break;
       case 0x42: Lcd_Out(1,1,txt);break;
      }
    }
  }
}
```

■dsPICF-300ボードの設定

このプログラムを正しく動作させるため、下記のようにdsPICF-300ボードの各種設定をセットしてください。ボードの設定が正しくないとプログラムは動作しませんので、設定内容とプログラムを比べて確認してください。設定としてはRS232C通信(USB-UART変換ポート)を使用できるようにすることと、ADコンバータのAN4を使用できるようにすることです。またLCDを使用します。

・すべてのディップスイッチはすべてOFF側に設定します。
・SW3の8番スイッチのLCDバックライトのみONに設定します。
・SW2の6番(RB1)と、8番(RB0)スイッチをONに設定します。
・ジャンパーJ15の"RB2"の位置をソケットでジャンパーします。
・LCDを取り付けます。

■パソコン側の準備

このプログラムでは、RS232C(UART通信)を行います。ただしここのチュートリアルでは、USB−UART変換IC(FT232R)を介してUART信号をパソコンのUSBポートでやりとりできる仮想COMポートを使います。(RS232Cポートは使いません。)

パソコンとの接続にはUSBケーブルが必要です。またパソコンと接続した際にはデバイスドライバーのインストールが必要となります。あらかじめドライバーをパソコンにインストールして認識できる状態にしておいてください。手順は本書11ページ・12ページの「⑲FT232RL搭載USB−UART変換ポート」項をお読み頂き、ドライバーをインストールして使える状態にしておいてください。インストールしたら、dsPICF−300の仮想COMポート番号をデバイスマネージャから確認しておきます。

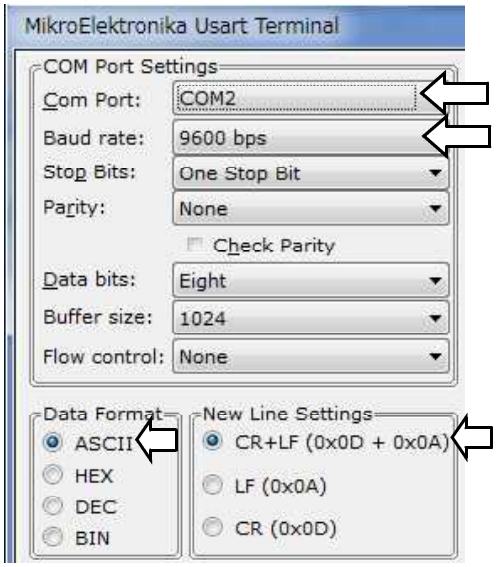
上図の例では、"COM5"に割り当てられていることがわかります。

RS232Cのターミナルソフトはmikro Cに付属しています。このターミナルソフトを使用すると大変便利に通信が行えます。

mikroCのツールバーにある"USART Terminal"のアイコンをクリックします。

ターミナルソフトが起動します。(アイコンが表示されていない場合には、メニューバーの"Tools"→"USART Terminal"で起動できます。)

通信に関する基本的な設定を行います。今回は、通信速度9600bps、1ストップビット、ノンパリティーの設定にしますので、次の図のように設定してください。"Com Port"は、dsPICF−300と接続したRS232Cのポート番号を指定します。設定が終わったら、"Connect"ボタンをクリックします。



■コンパイルと書き込みの実行

これまでのチュートリアルと同様、ソースプログラムをビルドして、HEXファイルをdsPICF−300のマイコンに書き込んでください。

■動作確認

ターミナルソフトの"Connect"ボタンを押すとRS232C通信が開始します。最初にADコンバータで取得した値を文字列として取得してみましょう。

ターミナルソフトの"Send"のテキストボックスに大文字で　A　と入力して"Send"ボタンをクリックします。

すると、その下の"Receive"ウインドウに値が表示されます。試しにP2ボリュームを回転させて、'A'を何回か送信してみましょう。回転させる毎に異なった値が返ることが確認できます。ボリュームを左に回しきった状態の時は0が返ります。右いっぱいに回しきると1000程度の値が返ります。本来10ビットのADコンバータなので最大値は1024ですが、ボリュームによって印加される電圧は最大でもVccレベル(3.3V)までは届かないので1000や1001あたりが最大値となります。10ビット分解能なので3.3/1024=0.00322…　なのでADコンバータで取得した値に0.00322を積算すると、実際の電圧を求めることができます。

同様にして、'B'も試してみましょう。文字Bを送信すると、ADコンバータで取得した値をLCDに表示します。

■プログラムの解説

void init(){

ユーザー定義の関数を作っています。関数名はinitで、voidはこの関数には戻り値がないvoid型であることを示しています。

C言語ではユーザーが任意の関数を作ることができます。予約後以外の文字列で関数を作りmain関数からコールできます。

ここではマイコンの初期設定に関する記述と、プログラム内で使用する組込関数の初期設定をこの関数内で行っています。

ある処理毎に関数を作ってmain関数からコールするようにすると、main関数内がすっきりとして見やすいプログラムを作ることができます。またデバッグの時もバグの特定や修正が容易になります。

Uart2_Init(9600);

UART2の初期化を行う関数です。()内には通信速度をbps単位で記述します。ここでは9600bpsに設定しています。

PIC24F16KA102には2系統のUARTがあります。本プログラムではLCDの制御ポートとの干渉やADコンバータ入力との干渉を避けるためUART2を使いました。

delay_ms(100);

UARTのユニットが初期化するのに約100ミリ秒かかりますので、100ミリ秒の遅延を挿入しています。delay_ms( )関数は、引数で指定した秒数(単位はミリ秒)だけ遅延を作ります。

char uart_rd, txt[7];

char型の変数1つと、要素数7の配列を作っています。

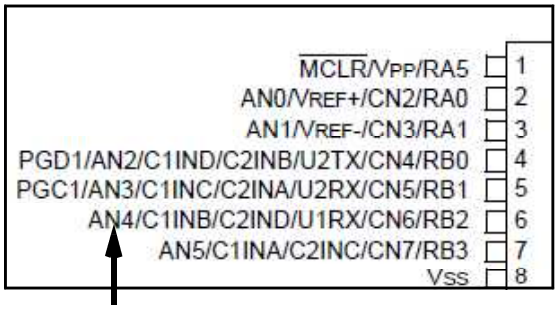
同じ型の場合には複数の変数名をカンマで区切って宣言できます。

init();

init関数をコールしています。init関数を実行した後には、ここに再度処理が戻ってきます。

AD_res = ADC1_Read(4);

Adc1_Read()関数は、( )内に指定したチャンネルからAD変換の値を取得して返します。10ビット分解能なので戻る値は0～1024までです。変数にはint型変数を使用しています。

> **チャンネル番号と、ピン番号は違います**
>
> ADコンバータのチャンネル番号と、PICのピン番号を混同してしまう方がいますが、チャンネル番号とピン番号は違います。
> ADコンバータのチャンネル番号はデータシートに記載があります。
>
> 
>
>
> 上図はデータシートからの抜粋です。AN4はRB2に割り当てられていることが分かります。

intTostr(AD_res,txt);

int型の変数を文字列として配列に代入する関数、intTostr()関数を使って文字列を作っています。

AD変換によって得られた値はint型変数に10ビット分解能で格納されますが、このままでは0～1024までの値なので、LCDに表示させたりUARTで文字列として送信させたりすることはできません。よってここでint型変数の内容を文字列に変換できる関数を使います。

intTostr(変換対象の変数 , 要素7の配列)という書式になります。この文字列として代入される配列は必ず要素数が7でないといけないという仕様上の決まりがあります。

if (Uart2_Data_Ready()) {

Uart2_Data_Ready()関数はUARTの受信バッファにデータが格納されると真になります。これをif文で検出することでデータの受信を監視します。

uart_rd = Uart2_Read()

char型の変数uart_rdに受信データを取り込みます。1バイトサイズのデータのみ受信できます。連続するデータの場合には、配列とポインタを使用します。

switch (uart_rd){

switch文は、( )内の評価式を評価してcaseで指定された値のルーチンに処理を分岐します。評価式にuart_rd変数を指定し、UARTで受信したデータの内容によって処理を分岐しています。

caseの後には値を記述し、評価式がその値になると、そこにジャンプします。ここでは、'A'及び'B'を指定することで、受信した文字によって処理を分岐しています。

C言語では、1文字の場合には ' (シングルクォーテーション)で囲います。(文字列はダブルクォーテーションです。)

なお、break文を記述すると、switch文から処理が抜け出します。

UART2_Write_Text(txt)

Uart2_Write_Text()関数は、( )内に記述した配列の内容をUARTで出力する関数です。文字列を出力したい場合には、文字列をそのままダブルクォーテーション(")で囲みます。

チュートリアル④　～TMR1を使用した割込プログラム～

■TMR1とは‥

PIC24シリーズのデバイスには全てのシリーズでタイマ1からタイマ5までの5つの16ビットタイマが内蔵されています。内蔵タイマのことをTMR1とかTMR2とか記載します。16ビットのタイマということは、0x0000～0xFFFF(0～65535)までをカウントするということになります。

5個のタイマの基本的な構成は同じですが少しずつ違いがあります。その中でTMR1の特徴は、外部クロックの場合として32.768KHzの水晶発振子を接続して使用できる発振回路が内蔵されていることが挙げられます。ここに32.768KHzの水晶発振子を接続すると、リアルタイムクロックとして使用できる仕様となっています。

TMR1は、16ビットのカウンターで、内部クロックか外部クロックに同期させてカウントを行わせることができます。カウンタはアップカウンタとなっていてパルス入力ごとに＋1されます。

TMR1は、PR1レジスタの値と常に比較されるようになっており、TMR1の値と、PR1レジスタの値が一致した時、割込が発生する仕組みになっています。また割込発生後は、TMR1の値は0に初期化されます。

■PIC24の割込について

PIC24では、割込の種類(何のイベントで発生した割込か)によって、ジャンプするベクタ(ジャンプベクタと呼びます)が異なるベクタ方式という割込を採用しています。そして、その割込に対応する処理ルーチンのことを割込サービスルーチンといい、ISRと呼びます。

8ビットのPICマイコンに比べ格段に構造が複雑になっています。これにより、様々な割込に応じて異なる割込ルーチンを作成でき、より複雑な割込プログラムを作ることができます。

ジャンプベクタは、プログラムメモリーのアドレスに割り当てられています。割込が発生すると、その割込の種類によって、ジャンプベクタが変わります。どんな割込が発生した時、どのジャンプベクタへジャンプするのかは決められており、例えば今回使用するTMR1割込の場合には、ベクタ番号は3となり、IVTアドレスは0x0001Aです。(この表はPICマイコンのデータシートに記載されています。)このように割込の種類と、飛び先アドレスの一覧を主ベクタテーブル(IVT)と呼びます。IVTは、プログラムメモリーの0x00014～0x00FEに配置されています。

IVTの他にもう1つベクタテーブルが用意されており、これを副ベクタテーブル(AIVT)と呼びます。これは、プログラムメモリーの0x00114～0x001FEに配置されています。これは、IVTが何らかの事情で使用できない場合に、IVTに変わってAIVTを使用することができることを意味します。基本的には、IVTを使用すると覚えておいてください。

また、PIC24では割込に順位が付くようになりました。これまでの8ビットのPICでは、割込は同時には1種類だけしか選択できませんでしたから、発生した割込の順位などは最初から必要ありませんでした。しかしPIC24では複数の割込をベクタ方式で処理できるため、割込に優先順位を持たせてあります。

優先順位は、レベル0～レベル15までの16段階です。15が最も優先順位が高い割込になります。TMR1割込のような汎用割込の場合には、レベル7～レベル0が設定されます。

では順位はどのように決まっているかというと、IVTの並び順となっています。IVTの値が小さい方が、レベルが高い、すなわち優先的に割込として受け付けられるということです。
例えば、TMR1の割込はIVTでベクタ番号3です。対してADコンバーターの読み取り完了時に発生する割込はベクタ番号13です。よって、2つの割込が同時に発生した場合には、TMR1の方が優先されるということになります。

さらにCPUには割込許可レベルというものがあり、これも0～15で設定します。これは、CORCONレジスタのビット3と、SRレジスタのビット7～5の計4ビット(CORCONレジスタのビット3が最上位ビット)で指定するもので、ここで設定した値よりも、優先順位が高い割込が、実際の割込として許可される仕組みになっています。

■プログラムの内容
TMR1割込を使用して、200ミリ秒毎に割込を発生させて、PORTBの出力を、200ミリ秒ごとに反転させるプログラムを作ります。

■プログラムの仕組み
今回は、dsPICF-300のボードに最初から装着されている8MHzの水晶発振子をクロックとして使用します。クロックは逓倍せずにそのままCPUのクロックとして8MHzとして使用します。
TMR1のクロック源は内部クロックとします。PIC内部では動作クロックが1/2になります。よって、内部は4MHzで動作しています。
時間に直すと1周期250nsです。タイマーの時間は下記の式で求められます。

TMR1時間=(1/2×動作クロック) × プリスケーラー分周比 × (PR1レジスタの設定値-1)

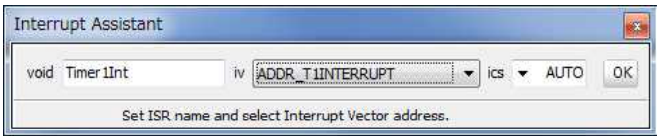
今回は下記の式により約200ミリ秒を作ります。

250ns×256×(3126-1) = 200ミリ秒

■プログラムの作成
新しいプロジェクトを作成して、プログラムを作ってください。動作周波数は8MHzに設定してください。
mikroCには、dsPICやPIC24用に割込関数を記述するためにサポートツールが用意されており、このツールを使うと割込のベクタ番号やIVTアドレスをユーザーが調べなくても割込の処理を記述することができます。今回はこのツールを使ってプログラムを作ってみましょう。
最初にmain関数に関する記述を今まで通り入力していきます。下記のようにプログラムを書いてください。

```
void main() {
  ADPCFG = 0xFFFF;
  TRISB = 0;
  LATB = 0;

  T1IF_bit = 0;
  T1IE_bit = 1;
  T1CON = 0x8030;
  IPC0 = 0x7000;
  PR1 = 3126;
}
```

main関数は、PICマイコンのレジスタの設定の記述だけになります。続いて、割込が発生した時に行う処理について記述をします。このチュートリアルではTMR1の値がPR1レジスタの値と同じになった時、割込が発生します。TMR1の割込関数を作るためにサポートツールを使います。プログラムの void main() { と記述されている部分の void の"v" の前に先頭部分にカーソルを置きます。
メニューバーの"Tools"→"Interrupt Assistant"をクリックして起動します。ツールが起動しますので、"void"の後ろのテキストボックスに、関数名を入力します。ここでは、"Timer1Int"と入力します。
続いて、"iv" の右側のプルダウンから"ADDR_T1INTERRUPT"を選択します。"ics" の右側のプルダウンは"AUTO"を選択します。

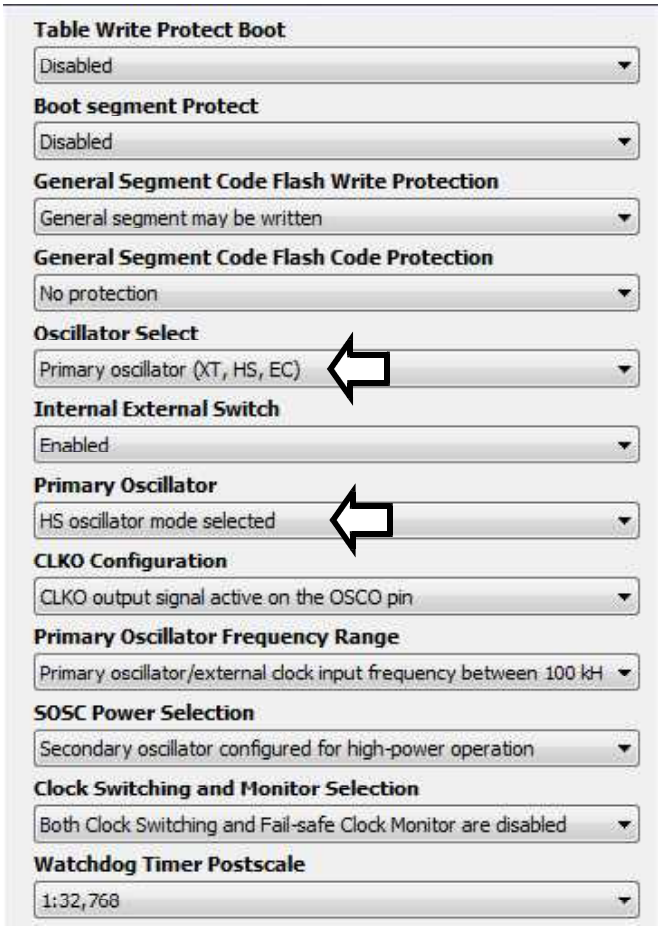


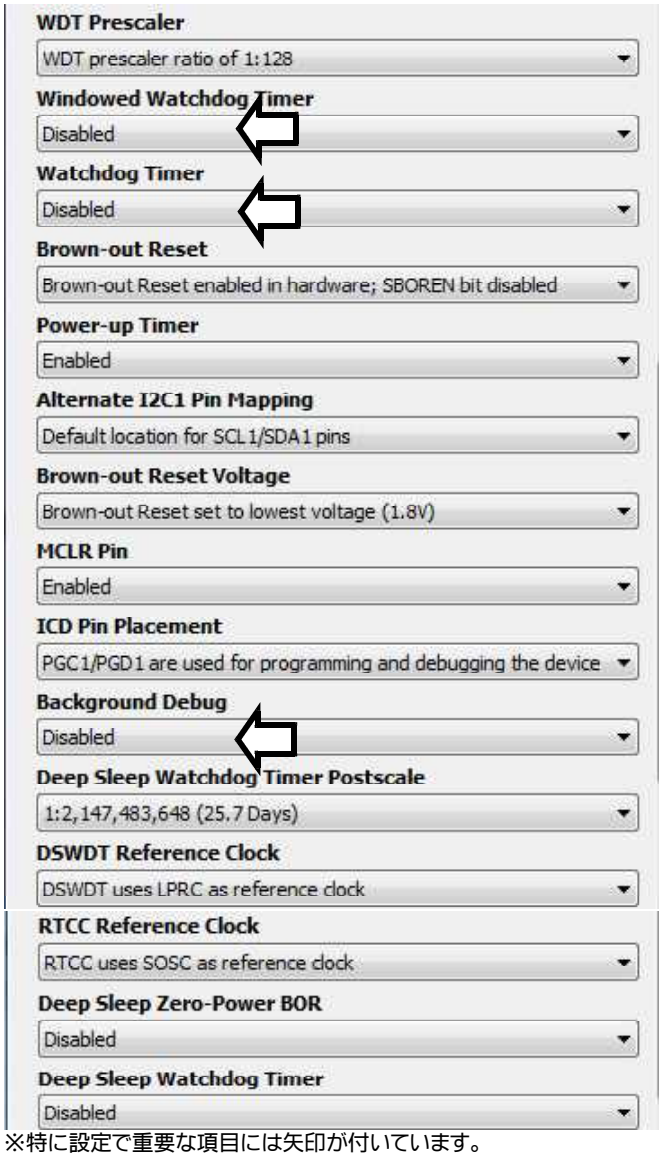
上図のように設定できたら"OK"ボタンを押します。すると、自動的に新しい関数 "void Timer1Int() iv IVT_ADDR_T1INTERRUPT ics ICS_AUTO { }"が作られます。この新しくできた関数の{ }までの間に割込発生時に行う処理を記述します。次のように記述します。

```
void Timer1Int() iv IVT_ADDR_T1INTERRUPT
ics ICS_AUTO
{

     T1IF_bit = 0;
     LATB = ~ LATB;
}
```

※上例では関数名が長いため2行になっていますが実際には1行です。

■コンフィギュレーションビットの設定
正しくプログラムが動作するようにコンフィギュレーションビットの値を設定しましょう。メニューバーの"Project"→"Edit Project"をクリックし、次のように設定します。





※特に設定で重要な項目には矢印が付いています。

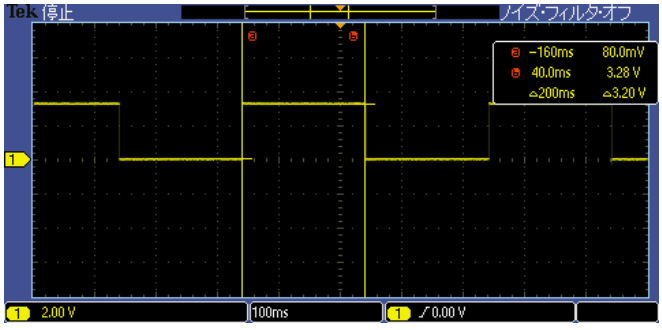
設定が完了したらOKボタンを押して閉じます。コンパイルして書き込みを実行してください。

■dsPICF-300ボードの設定

SW4の"PORTB"のスイッチだけをONにしてその他はすべてOFFに設定します。LCDは外しておいてください。

■動作確認

プログラムを書き込むと、PORTBに接続されたLEDが全部200ミリ秒間隔で点滅します。オシロスコープをお持ちであれば、PORTBのいずれかのピンの波形を見ると、200ミリ秒間隔の矩形波を観察できます。



■プログラムの解説

void Timer1Int() iv IVT_ADDR_T1INTERRUPT ics ICS_AUTO{

TMR1の割込が発生した時に実行されるルーチンです。
mikroCでは、ivキーワードがあり、ivキーワードに続いて、割り込みの種類を指定することで、その割り込みが発生した場合に、{ }内に記述されたルーチンが実行されます。
通常は、main関数が実行されていますが、割り込みが発生した時にこの関数が実行されます。

T1IF_bit = 0;

TMR1割込発生フラグを初期化しています。IFS0レジスタのビット3に0を代入しています。割込先ルーチンでは、必ず対応する割込発生フラグを初期化しないと、次の割込を発生させられませんのでここで初期化しています。
mikroCでは、レジスタの特定のビットを指定する場合には、"ビット名_bit"で指定することができ、代入演算子で値を代入できます。

LATB =～ LATB;

ポートの状態を反転させています。"～"演算子(チルダ)はビット演算のNOT演算子です。LATBの状態をNOT演算子で反転させて代入することで点滅させています。

T1IE_bit = 1;

IEC0レジスタのビット3にある、TMR1割込の許可制御ビットを1にしてTMR1の割込を有効にしています。

T1CON = 0x8030;

T1CONレジスタは、TMR1の各種設定をするためのレジスタです。
ここでは、「外部ゲートOFF・プリスケーラー分周比256・クロック源は内部クロック」という設定にしています。

IPC0 = 0x7000;

IPC0レジスタでは、割込の優先順位を制御します。
ビット14～11がTMR1の割込レベルの設定ビットになっており、000で禁止、111でレベル7(最高)となります。
今回は、TMR1割込しか使いませんので、最高レベルの111に設定しています。

PR1 = 3126;

PR1レジスタの値です。TMR1は常にここの値と比較され、ここの値と一致した時に割込を発生させます。
今回は前のページに記載した計算によって求めた値を代入しています。

チュートリアル⑤　～静電容量式タッチスイッチを使う～

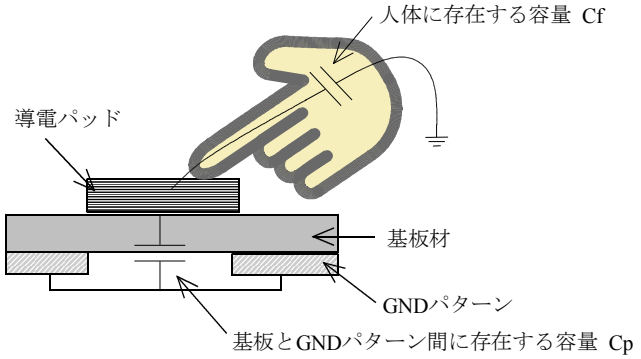
PIC24F16KA102には、静電容量方式でタッチスイッチを作るために必要なCTMU(Charge Time Measurement Unit)というモジュールが内蔵されています。このモジュールを利用すると、基板上にパッドを形成するだけで、タッチスイッチが作れます。

dsPICF-300ボードにはこの静電容量式のタッチスイッチとして利用できるパッドが用意されています。このチュートリアルではこのパッドを使った静電容量方式のタッチスイッチを作ってみましょう。

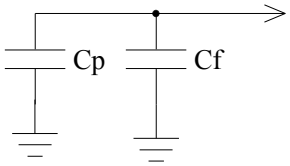
■静電容量方式タッチスイッチの原理

静電容量方式というくらいですから、コンデンサの原理を利用します。人体には豊富な水分があり、それ全体が大きなコンデンサと見立てることができます。指などで金属など電流が流れるものに触れると、人体という大きなコンデンサがそこに接続されたような状態となります。

「人体は大きなコンデンサ」この原理を利用します。



原理としては上図のようになります。基板にはGNDパターンと導電パターンの間には必ず容量Cpが存在しています。ここに人体の一部が触れると、人体に存在する容量Cfが並列接続された状態になります。



もともとあった容量成分Cpに、人体の容量成分Cfが接続されることで浮遊容量が変化します。これを充電電圧によって違い検出し、タッチセンサとして使うのが、PICで実現する静電容量方式のタッチスイッチです。なお人体の静電容量は約100pF～670pF程度です。

■CTMUモジュールの役割とタッチ判定の原理

CTMUモジュールの役割は上図はCpとCfの容量に対して充放電を定期間隔でさせることにあります。実際の読み取りはADコンバータで行うので2つのモジュールが協調して動作することになります。

CTMUモジュールでは、まず最初に基板そのものに存在するCpや、ADコンバータにある容量成分などすべてを並列にした時の電荷を放電させます。

放電が完了したら、一定の電流を流して容量を充電していきます。容量に充電していきますので、だんだんと電圧は上昇していきます。一定時間が経過したら充電を停止して、ADコンバータでその時の電圧の値を読みます。この時の電圧値は、充電時間と流した電流に比例し容量に反比例します。(下式)

$$V = \frac{I \times t}{C}$$

※ V:充電後の電圧値、I:充電電流、t:充電時間、C:全体の容量

このことから、容量が大きいと電圧は下がることがわかります。指が触れた時はCpとCfが並列接続となるため容量が増加しますで、一定時間充電した後の充電電圧は下がることになります。この電位差によって、タッチの有無を検出することになります。

流す電流の大きさはCTMUモジュールの設定で決めることができます。設定値は0.55uA(×1設定)、5.5uA(×10設定)、55uA(×100設定)の3段階です。充電時間はプログラムで作ります。

ADコンバータは10ビット分解能なのでVddが3.3Vとすると、0.00322V程度の違いを判定できますので、これらを協調させて動作させれば、判定精度をかなり高くすることができます。

■プログラムを考えてみよう

このチュートリアルではdsPICF-300のタッチパッドを使って、RB0に接続された▼印のタッチパッドに触った時、LCDに"Touch Finger!"と表示されるプログラムを作ります。触っていない時には、"No Finger"と表示させます。また、電圧値の変動を観察するためLCDの2行目には取得した電圧値の平均値を表示させることにします。

■プログラムの仕組みを考えてみよう

このプログラムの仕組み、アルゴリズムはマイクロチップから提供されているリファレンスの内容に従って作ります。基本的なプログラムの仕組みは次のようになります。

①すべてのピンをデジタルI/Oピンにして一度論理ゼロ(L)にセットする。これによって静電容量式タッチセンサパッドを全て接地してその電位を0Vにする。5ミリ秒待つ。
②読み出すA/Dチャンネルをアナログ入力に設定する。
③選択したチャンネルにA/Dコンバータを設定する。
④CTMUCONレジスタの放電ビットIDISSENを1にセットする。
　これによりA/Dコンバータの内部静電容量を電位0Vにする。
⑤電荷が放電されるまで待機する。5ミリ秒程度待機。
⑥放電ビットIDISSENをクリア(0にする)する。放電終了。
⑦ADコンバータのサンプルビット、SAMP（AD1CON1)をセットして、選択したチャンネルのサンプリングを開始する。
⑧CTMUのエッジステータスビットのEDG2STATビットを0にする。
　→トリガリセット
⑨EDG1STATビットを1にセットする。これにより、選択されたA/D入力に対するCTMU電流源がONになり、充電を開始する。
⑩一定の時間充電するため待機する。
　この時間によって電圧値が変わる。設定値は2～20マイクロ秒程度。
⑪上記でセットしたエッジステータスビットをクリア。これにより、CTMU電流源がOFFになって充電が終了する。
⑫選択したチャンネルのA/D変換を開始。
⑬変換が完了するまで待機。
⑭A/Dコンバータによって読み出された値を変数に格納する。

上記の14ステップを実行して、A/Dコンバータの値を変数に取得します。但しアナログ電圧を1回取得しただけでは変動があり正確さに欠けるので、5回取得して、その平均値を使うことにします。

また、基準となるタッチしていない時の電圧値はその時々(周囲の湿度や環境等)や基板材料等によって相当変わるため、最初にタッチしていない時の基準電圧を求めておき、その基準電圧から、ある程度電圧が変動した所に閾値を設定し、その閾値を超えた時点でタッチしたことを判定するようにプログラムを作ります。

プログラムは、設定項目が多くやや難しいです。お手元にPIC24F16KA102のデータシートをご用意の上、CTMUのページとADコンバータのページをそれぞれ参照しながら、1行ずつ意味を確認してプログラムしてみてください。

```
sbit LCD_RS at LATB4_bit;
sbit LCD_EN at LATB5_bit;
sbit LCD_D4 at LATB6_bit;
sbit LCD_D5 at LATA0_bit;
sbit LCD_D6 at LATA1_bit;
sbit LCD_D7 at LATA4_bit;

sbit LCD_RS_Direction at TRISB4_bit;
sbit LCD_EN_Direction at TRISB5_bit;
sbit LCD_D4_Direction at TRISB6_bit;
sbit LCD_D5_Direction at TRISA0_bit;
sbit LCD_D6_Direction at TRISA1_bit;
sbit LCD_D7_Direction at TRISA4_bit;

unsigned short cont=0;
int average=0, add_num=0, Vreg=0;
char txt[7];

void init(){

    //CTMUCON
    CTMUCON = 0x124;

    //CTMUICON
    CTMUICON = 0x300;               //55uA IRNG1:IRNG0<1:1>

    //ADC Setup
    AD1PCFG = 0;                    //All pin are analog
    AD1CON1 = 0;
    AD1CHS = 0;                     //select the analog channel0
    AD1CSSL = 0;
    AD1CON1bits.FORM = 0;           //Unsigned fractional format
    AD1CON3 = 0;                    //bits.ADRC=0;
    AD1CON2 = 0;
    AD1CON1bits.ADON = 1;           //Turn On A/D
    CTMUCONbits.CTMUEN = 1;         //Enable CTMU

    Lcd_Init();
    Lcd_Cmd(_LCD_CLEAR);
    Lcd_Cmd(_LCD_CURSOR_OFF);
}
int Get_Value(){
    int ad_val=0;

     AD1PCFG = 0xFFFF;              //set all A/D channels to digital I/O pins
     TRISB = 0;
     LATB = 0;
     PORTB = 0;
     delay_ms(5);                   //Wait for charge to drain 5mS

     TRISB = 0x01;                  //set input  RB0=AN2
     AD1PCFG = 0xFFFE;              //RB0 is analog input pin
     delay_ms(5);                   //wait for A/D to connect to channel

     AD1CHS = 0x002;                //select A/D channel analog CH2
     CTMUCONbits.IDISSEN = 1;       //Drain any charge on the A/D circuit
     delay_ms(5);                   //Wait for charge to drain
```

```
    CTMUCONbits.IDISSEN = 0;        //Stop discharge of A/D circuit
    IFS0bits.AD1IF = 0;             //Make sure A/D interrupt flag = 0
    AD1CON1bits.SAMP = 1;           //Manually start sampling
    CTMUCONbits.EDG2STAT = 0;       // Make sure edge2 is 0
    CTMUCONbits.EDG1STAT = 1;       // Set edge1 - Start Charge

    delay_us(5);                    // Delay for CTMU charge time (5microSec)

    CTMUCONbits.EDG1STAT = 0;       //Clear edge1 - Stop Charge
    IFS0bits.AD1IF = 0;

    AD1CON1bits.SAMP = 0;           //Clear SAMP bit to begin manual A/D conversion
    while(!IFS0bits.AD1IF);         //Wait for the A/D conversion to finish
    ad_val = ADC1BUF0;              //Read the value from the A/D conversion

    AD1CON1bits.SAMP = 0;
    IFS0bits.AD1IF = 0;
    AD1CON1bits.DONE = 0;

    return(ad_val);
}

void main() {
    unsigned char n;

    init();

    while(1){
        for (n=0; n<5; n++){        //Get value 5times
            add_num += Get_value();
            delay_ms(10);
        }
        average = add_num / 5;
        add_num=0;

        Vreg = 300;                 //Set threshold value (e.x. 300)

        if(Vreg < average){
            Lcd_Out(1,1,"Touch Finger!   ");
        }else{
            Lcd_Out(1,1,"No Finger       ");
        }

        inttostr(average,txt);
        Lcd_Out(2,1,txt);
    }
}
```

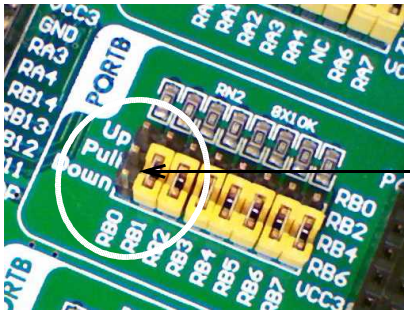
動作周波数は8MHzとしてプログラムを作ってください。コンフィギュレーションビットの設定は、チュートリアル4と同じです。

■ボード設定

プログラムを書き込む前にボードの設定をしておきましょう。ボードの設定が正しくないとプログラムは正しく動きません。

・16×2LCDを取り付けてください。

・PORTBのプルアップ/ダウン設定ジャンパの"RB0"のソケットを外してください。



ソケットを外す

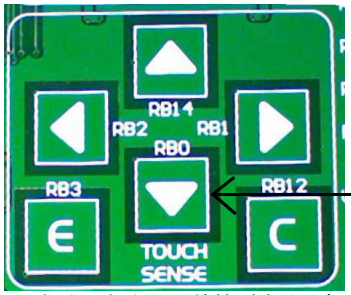
・その他のジャンパーソケット、ディップスイッチはすべてOFF位置にしてください。特にJ15ジャンパーはRB0の所にソケットが入っていないことを確認してください。

プログラムが完成したらビルドをして書き込みを行ってください。

■動作確認と調整

プログラムを書き込むと、LCDの1行目には"No Finger"と表示されます。2行目には数字が表示されます。値はおおむね150～400程度の値をしめしているはずです。もし値がずれている場合には、ボードの設定が間違っているか、プログラムが正しい動作周波数設定で動作していない可能性があります。(プログラムは8MHz動作です。)

ここでボードのタッチパッドの『▼』(RB0)のパッドに軽く振れてみてください。



指で直接触れる

パッドに振れると数値が大きく変わる(LCD2行目の数値)ことを確認します。同時にLCD1行目の表示が"Touch Finger"と表示されることを確認します。

もし正しくタッチを検出しない場合には、最初に何も触っていない状態の時の値を、紙などに書いて記録しておきます。数値は多少変動がありますが、だいたいの値を書いておきます。この値が指を検出していない時の値です。(例としてここでは230とします。)

続いてパッドにタッチしてみます。値が変わりますので、その時の値を記録します。値はめまぐるしく変動しますが、だいたいどのくらいの値かを確認しておきます。この値がタッチした時の値です。(例としてここでは500位とします。)

この2つの値の中間くらいに閾値を設定します。この値未満ならば指は触れていない、この値以上ならば指が触れている‥と判断する値です。例えば上記の例だと非接触時230、接触時500位なので、閾値を300に設定します。この値を記述しているのがプログラム後半にある『Vreg = 300; 』の文です。もしタッチ検出の精度がおかしい場合には、この値を調整してタッチ検出の精度が高くなるようにしてください。

※他のパッドについて動作させたい場合には、RB1,RB2,RB3,RB12,RB14についてそれぞれ周期的にAD1CHSレジスタの値を変えながら、アナログ入力電圧値を読むピンを変更しながら値を取得する構造にします。取得した電圧値は5回の平均値をとり、あらかじめ設定した閾値と比較することで、タッチの検出を行うようにプログラムします。

チュートリアル⑥　～ICD機能を使ったデバッグを体験する～

■ICD機能とは?

ICDとはインサーキットデバッグの略で、ターゲットボード上のデバイスに実際にプログラムを書き込み、動作させながらパソコン上でデバッグを行う手法のことです。

例えば現在実行している行をハイライト表示させながら実機での動作が見られますので、プログラムの挙動を簡単に把握できます。また、1行ずつプログラムを実行したり、変数やレジスタの現在の値をリアルタイムに閲覧することができたりします。

本セットに付属のmikroCには、C言語レベルでデバッグのできるICD機能が搭載されています。dsPICF-300と組み合わせることで、ICD機能を簡単に体験することができます。

■ICD機能を使うには

ICD機能を使用する場合、プログラムには特に大きな変更点はありませんが、delay_(ms)などのdelay関数が入ったプログラムは正しく動作しません。ICD機能を使用する場合、コンパイル時に"Project Settings"から、"Build type"を"ICD debug"設定にする必要があります。このチュートリアルを通してICDの使い方を一通り体験してみましょう。

■プログラムの内容

LCDに文字"microtechnica"を表示されます。但しICD機能の動作を確認するため1文字ずつ表示するプログラムを作ります。

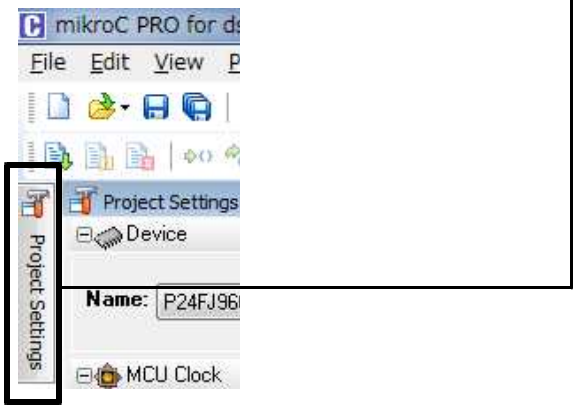
■プログラムの作成

新しいプロジェクトを作成して、プログラムを作ってください。
プロジェクトの作成方法等は従来の方法と全く違いはありません。

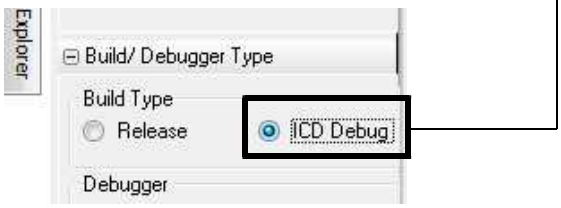
```
sbit LCD_RS at LATB4_bit;
sbit LCD_EN at LATB5_bit;
sbit LCD_D4 at LATB6_bit;
sbit LCD_D5 at LATA0_bit;
sbit LCD_D6 at LATA1_bit;
sbit LCD_D7 at LATA4_bit;

sbit LCD_RS_Direction at TRISB4_bit;
sbit LCD_EN_Direction at TRISB5_bit;
sbit LCD_D4_Direction at TRISB6_bit;
sbit LCD_D5_Direction at TRISA0_bit;
sbit LCD_D6_Direction at TRISA1_bit;
sbit LCD_D7_Direction at TRISA4_bit;

void main(){

   char text[14]="microtechnica";
   int i;

   ADPCFG = 0xFFFF;
   Lcd_Init();
   Lcd_Cmd(_LCD_CLEAR);
   Lcd_Cmd(_LCD_CURSOR_OFF);

   for(i=0 ; i<13 ; i++){
      Lcd_Chr_Cp(text[i]);
   }
}
```

■コンパイルと書き込み

プログラムが書き終わったら、コンパイルをしますが、この時従来とは違い、Build typeをICD用に変更します。画面左下の"Project Settings"タブをクリックします。ウインドウが展開します。



ウインドウが表示されたら、その中の"Build/Debugger Type"の部分にある"Build Type"のチェックボックスのうち、"ICD debug"にチェックを入れます。



チェックを入れたら、従来通り、メニューバーのビルドして、コンパイルが終了したら従来どおり、PICマイコンへ書き込みを行ってください。この作業はこれまでと同じです。

"ICD debug"オプションを指定してコンパイルしたプログラムは、PICマイコンへ書き込んだ後にすぐには動作しません。

■動作の前にボードの確認をしましょう

実際にプログラムを走らせる前には、必ずボードの設定状態を確認してください。ボードの設定が正しくないと動作しないことがあります。

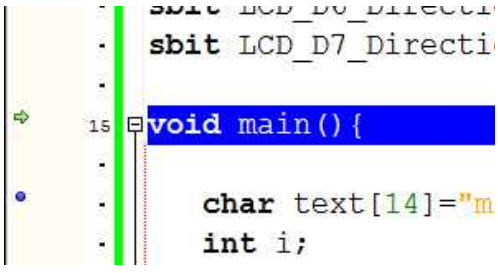
・LCDを取り付けてください

■ICDを動作させてみる

ICDを実行して、プログラムの動作内容を確認しましょう。
LCDに何も文字列が表示が出ていないことを確認します。文字列が表示されている場合には一度dsPICF-300の電源を切断してLCDの画面をクリアした状態で下記を実行してください。

1 メニューバーの"Run"→"Start Debugger"をクリックします。
キーボードの"F9"キーでも同様の操作ができます。

2 void main() { の行が青色バーでハイライトされます。
ICDでは実行される行が青色バーでハイライト表示されます。
また、行数表示の左隣に➡マークが表示され、実行している行が分かります。



※void main() { の行が青くハイライトされず、別のアセンブラの画面が表示されてしまった場合には、メニューバーの"Run"→"DisAssembly mode"をクリックしてください。

3 ICDでは、1行ずつ実行する場合、LCD関連のライブラリーは動作が複雑で時間がかかるため、ここでは一気にfor文の所までは通常通りプログラムを動作させてしまいましょう。

"for(i=0 ; i<13 ; i++){ " の行をクリックします。カーソルがこの行に移動していることを確認してください。

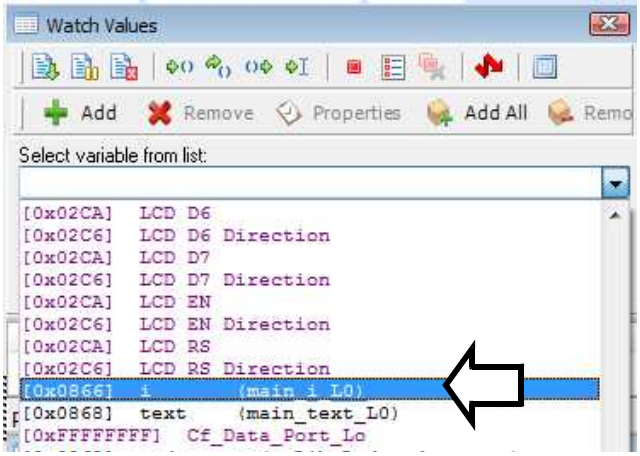
4 カーソル行までプログラムを一気に動かす方法でfor文位置までプログラムを移動させます。
メニューバーの"Run"→"Run to Cusor"をクリックします。キーボードのF4キーでも同様の操作ができます。
for文の一行が青色バーでハイライト表示されます。

5 LCDにはまだ文字は表示されていないはずです。
この状態で1行ずつ実行するステップ実行を行い、1文字ずつLCDに文字が表示されるのを確認しましょう。
キーボードのF8キーを押します。又はメニューバーの"Run"→"Step Over"をクリックします。F8キーを押すと、青色バーでハイライト表示された部分が移動します。
F8キーを2回押すと、LCDに"m"と表示されます。

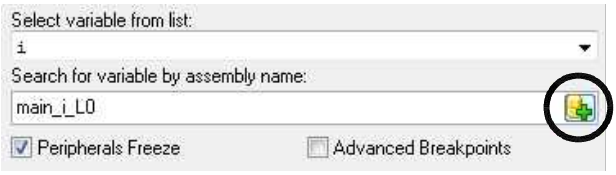
6 同様にして、F8キーを何回も押すと、プログラムはFor文を繰り返し実行して、1文字ずつLCDに文字が表示されていきます。5文字程度表示されたら次の手順を実行します。
なお、F8キーを早く押してしまうと、ICD機能の処理が間に合わずエラーが表示されてしまうことがあります。F8キーを押してステップ実行を進める場合には、必ず青い実行行を示すバーが次の行に移動してからF8キーを押すようにしてください。

7 ICDを実行すると、Watch Valueウインドウが画面の右側に表示されます。ウォッチウインドウは、変数やレジスタの値をリアルタイムで閲覧できるウインドウです。ここでは変数iの値が1ずつ増えていく様子を確認しましょう。

8 変数iを追加します。
"Select Variable form list"のプルダウンをクリックして、その中から "[0xXXXX] i (main_i_LO)" と表示されている部分クリックします。※XXXXは値が入ります。



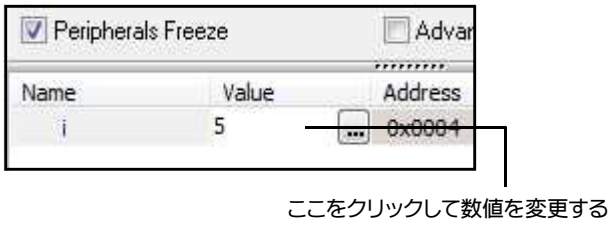
9 Addボタンを押して、追加します。



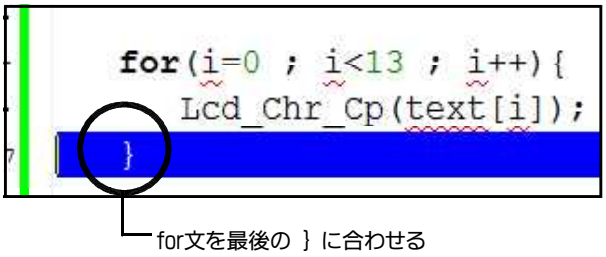
10 リストに追加されるとNameとValue、Addressが表示されます。
Valueの値は現在の変数iに格納されている数値です。
F8キーを押してFor文を実行すると、1ずつiの値が増えていく様子を見ることができます。

値が更新されると数値は赤文字で表示されます。

11 また、この"value"の値を手動で違う値にするとリアルタイムで、変数の値を変更してしまうことができます。
"value"のところの数字をクリックすると、数値が変更できるようになりますので、ここに値を10進数で入力します。



最初にステップ実行を何回か行って、青い実行行を示すバーをfor文の最後の所の } にしておきます。



12 ここでは値を"0"に変更してみましょう。

13 ステップ実行を行ってみましょう。F8キーを何回か押してステップ実行を行うと、LCDに文字"m"が表示されます。これは文字列"micro technica"の最初の文字"m"が表示されたためです。
for文のループカウンタが0に戻るため、表示される文字の位置が変わったためです。
このようにmikroCではC言語レベルで自在なICD機能を実行することができます。

■ICD機能を使用しない場合の注意

ICD機能を使用する場合、コンパイル時のBuild typeをICD用に設定しました。このICD用に設定されて生成されたHEXファイルは、通常動作はしません。ICD機能を使用しない通常のHEXファイルをビルドしたい場合には必ずビルドする前に"Project Settings"の"Build type"から"Release"にチェックを入れて、ビルドを行ってください。